12-2016

A Genetic Algorithm Incorporating Design Choice for the Preliminary Design of Unmanned Aerial Vehicles

Kenneth Michael Mull
Western Michigan University, ken.m.mull@gmail.com

A GENETIC ALGORITHM INCORPORATING DESIGN CHOICE FOR THE PRELIMINARY DESIGN OF UNMANNED AERIAL VEHICLES

by

Kenneth Michael Mull

A thesis submitted to the faculty of the Graduate College
in partial fulfillment of the requirements
for the degree of Master of Science
Mechanical and Aerospace Engineering
Western Michigan University
December 2016

A GENETIC ALGORITHM INCORPORATING DESIGN CHOICE FOR THE PRELIMINARY DESIGN OF UNMANNED AERIAL VEHICLES

Kenneth Michael Mull, M.S.

Western Michigan University, 2016

Unmanned Aerial Vehicles (UAVs) are currently at the forefront of aerospace technologies. The design of these aircraft is complex and often performance characteristics are coupled to multiple design attributes. At the early design phase both discrete and continuous design choices are present limiting the feasibility of traditional derivative based optimization techniques. In place of these methods, the design space can be explored using a genetic algorithm that mimics the process of natural selection, providing a capable and reliable base airframe constructed from the required performance metrics. By incorporating a genetic multidisciplinary optimization algorithm early in the conceptual design phase, aircraft can be moved faster and more cost effectively through the product development cycle thus reducing research and development costs, the time necessary to deliver a finished product, and the program and unit costs, while delivering a vehicle with superior performance characteristics.

TABLE OF CONTENTS

Table of Contents – Continued

Table of Contents – Continued

Table of Contents – Continued

LIST OF TABLES

List of Tables - Continued

LIST OF FIGURES

List of Figures – Continued

List of Figures – Continued

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.1.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles, (UAVs), represent one of the fastest growing fields in aerospace engineering. UAVs, also referred to as drones, have been around for several decades. Militaries around the world field low to middle range reconnaissance aircraft, and NASA uses a modified Global Hawk, to track and monitor hurricanes and other severe weather. UAVs comprise thousands of parts, incorporating the cutting edge of current technologies. This in turn necessitates large research and development operations, various production facilities and techniques, as well as skilled engineers, machinists, and administrative personnel to contribute to and oversee the development cycle. While these aircraft serve in both civilian and military capacities, the airframes with military roles represent the majority of existing designs and will continue to see heavy military investment in the future [3].

UAVs come in all shapes and sizes. They range from small radio controlled hobby planes to aircraft that rival commercial transports in dimension. Current UAVs can be simply grouped into fixed wing aircraft, and rotorcraft. Rotorcraft represent a large share of small scale UAVs but few exist outside this range. These small craft include the small commercial multirotors that have become common over the last decade. Fixed wing aircraft currently dominate the military market share, representing the majority of large scale UAVs. The MQ-8B Fire Scout unmanned helicopter represents the exception to this small scale limitation.

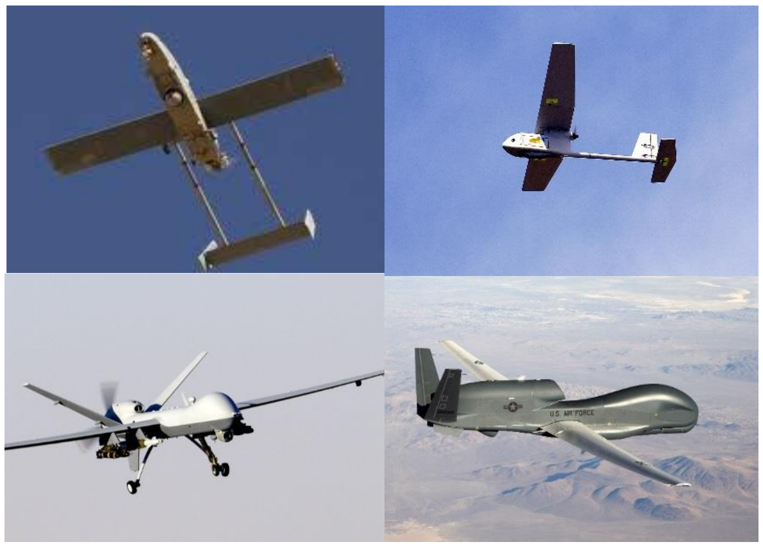


Figure 1.1 Military Unmanned Aerial Vehicles

Unmanned Aerial Vehicles currently used by the United States Military [17]. (*Top Left) RQ-2A Pioneer Short Range UAV (Top Right) Raven hand launched UAV [21] (Bottom Left) MQ-9 Reaper Combat UAV [20] (Bottom Right) RQ-4 Global Hawk Long Range UAV [16]*

1.1.2 Unmanned Aerial Vehicle Groups

UAV's are classified into various groups by the United States Department of Defense. The Defense Department uses the broad classification Unmanned Aerial Systems (UASs), when referring to UAVs. As illustrated in Figure 1.1 the military fields multiple unmanned systems, ranging in size from the tiny handheld launched “Raven” to the “Global Hawk”. Unmanned Aerial Vehicle groups are formed by considering the aircraft’s weight, altitude of operation, and maximum speed. There are five of these groups with Group 1 comprising the smallest, lowest flying, and slowest; and Group 5 containing the largest, highest flying and fastest.

Table 1.1 Department of Defense UAS Classifications [17]

UAS Groups	Maximum Weight (lbs)	Normal Operating Altitude	Speed (knots)	Representative UAS	
Group 1	0-20	<1200 AGL (1200 ft)	100	*Raven* WASP	
Group 2	21-55	<3500 AGL (3500 ft)	< 250	*ScanEagle*	
Group 3	<1320	<FL 180 (18000 ft)		*Shadow*	
Group 4	>1320		Any Airspeed	Fire Scout *Predator* Sky Warrior	
Group 5		>FL 180 (18000 ft)		Reaper *Global Hawk* BAMS	

1.2 Literature Review

1.2.1 Multi-Disciplinary Optimization

Optimization techniques find many suitable targets in engineering fields [2]. They can be applied at any part of product development from early preliminary stages to design tradeoffs before beginning a production run. The subject of an optimization can be as small as a single part or encompass a whole system. Large engineering projects often cross disciplines into other fields. The intricacies and often nonlinear dependencies between disciplines often determine the success or failure of a design. To successfully evaluate these systems Multi-Disciplinary Optimization (MDO) techniques can be applied to an objective or cost function that is subject to certain constraints function in order to select a design that performs optimally in two or more disciplines.

There are several types of optimization architecture. Martins [9]. MDO architectures are often classified by the order of the method used. First order optimization requires the computation of the first derivative of the objective function for use in predicting the increase or decrease of the objective function score for a selected change in design variables. Second order methods include the use a computed second order derivative to achieve a higher degree of accuracy with more complicated objective functions or to increase the rate of convergence. First and second order methods tend to converge more quickly as the derivatives provide a “best direction” to advance through the design space. Finally, zeroth order methods, often referred to as metaheuristic searches, more commonly utilized in a subclass called genetic algorithms (GA’s), explore the design space without the calculation of derivatives. Elbeltagi, Emad, Tarek Hegazy, and Donald Grierson explore five such algorithms in [5]. This class of algorithm is useful where derivatives of an objective function are not easily estimated.

1.2.2 Optimization for Aircraft

Aircraft are excellent targets for MDO architectures. The complex nature of aircraft design, including performance dependent on structure, thermodynamics, aerodynamics, stability and control, and manufacturing concerns, makes the aircraft as a whole or even a single part difficult to balance outside of an iterative design process. The expense of even a single part, or the loss of performance from a suboptimal design is driving the increased use of varied MDO architectures specifically for aircraft and aircraft subsystems. The majority of optimization takes place on existing designs, that is to say the product is already in a desirable form. As such derivative methods are faster [9] and more applicable to this later stage in the product development cycle. Common MDO routines have been applied to the aero-structural optimization of wings, cruise speed versus fuel consumption studies for commercial aircraft, stability and control surface optimization, airflow and thermal efficiency of a jet engine [9].

An example of an aero-structural optimization routine using a gradient based method can be seen in [7] where a high-fidelity model was applied to a proposed wide body commercial aircraft. While providing some guidance on how to decompose the design space particularly the variables that wing planform. This level of optimization routine requires an existing starting point. As such it is more suited to continued development of an aircraft design rather than an early preliminary design phase.

Derivative based optimization methods were also applied to the small scale UAV in [8]. The small electric cargo UAV had a prescribed structure and then a twist and taper aerodynamic optimization routine was utilized to maximize the lift over drag ratio. The Reynolds number regime of this small aircraft is significantly different from the designs explored within this document, limiting the incorporation of concepts to the architecture. The mission type constraints proposed for this electric UAV including weight, takeoff distance, stall speed, and operational payload however were adaptable to the genetic algorithm formulated within this document.

A stealth UAV optimization utilizing a genetic algorithm is described in [16]. Though the major layout of the flying wing aircraft was predetermined. The genetic algorithm actively searched a limited design space for the stealthiest design. The incorporation of radar cross section (RCS) as a performance variable within this research directly inspired its inclusion in this optimization routine.

A holistic aircraft design utilizing several types genetic algorithms is described by Raymer [13]. The Breeder Pool method in particular served as the starting point for the optimization routine developed during this research. Though the configuration of the aircraft was again predetermined before an optimization process took place, the vast majority of the constraints and the use of cost as an objective function can be directly associated with the developments made by Raymer.

1.2.3 Genetic Algorithms for Optimization

Genetic algorithms, being a zeroth order method, do not utilize derivatives in the iterative process. Instead genetic algorithms explore the design space by mimicking the processes of evolution and natural selection. A design string of variables is selected, and a population of design strings is rated on how well it performs in an objective function. The objective function acts as a test, similar to how an environment tests and ultimately determines the success of the organisms that live within it. The most successful designs have a high probability of good objective function scores, survival, and thus a high probability of passing its traits to the next generation. The poor designs are killed off or have a low probability of surviving to pass their traits to the next generation. Many genetic algorithms utilize elitism, mutation, and crossover routines when breeding two design strings together, Mutation and crossover encourage diversity within a population, exploring all facets of the design space for an advantage. Elitism guarantees a previously successful design is not lost when transitioning between generations, by immigrating the design(s) directly into the next generation.

Mutations involving a genetic algorithm are prescribed randomly to occur at a designed frequency. Mutations can be the sole driver of an optimization routine or used in conjunction with crossover operations. A pure mutation algorithm may have a single mutation or multiple mutations in each design generation. In order for a mutation to occur a random place on the design string is selected. Then the selected gene is perturbed by some specified amount. This mutated design now proceeds into the new generation. A single mutation operation is shown in Figure 1.2.

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{bmatrix} \rightarrow \begin{bmatrix} B \\ C \\ D + \delta\ mutation \\ E \\ F \\ G \end{bmatrix}$$

Figure 1.2 Single Mutation Operation

Crossover operations split the design string of two selected designs and append them to one another. First two parent designs are selected from the current generation. Second a specified or randomly selected point on the design string is selected. The parent strings are then split into four separate design strings. In example parent A and parent B are split at position three in the design string. Two offspring are created from the recombination of the split design strings. The top of string A is combined to the bottom of string B and the bottom of string A is appended to the top of string B. This process can be seen in Figure 1.3.

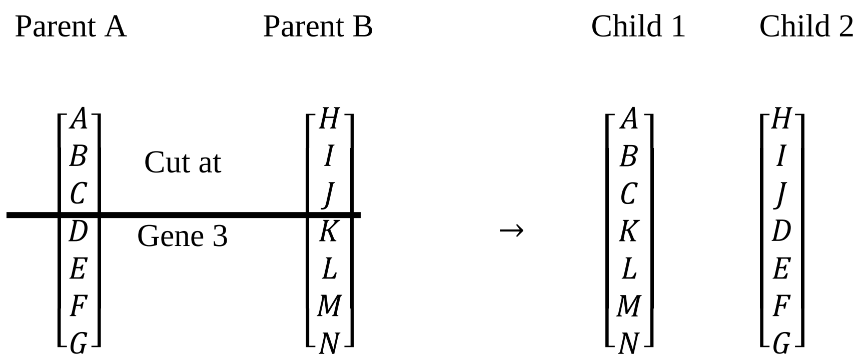


Figure 1.3 Crossover Operation

The algorithms methods are named depending on how the parents are selected and by extension how the next generation is created. The Killer Queen algorithm is a hyper elitist strategy relying solely on large quantities of mutations to generate the new population. It can be seen naturally in insect colonies [5] [13] where a single queen is responsible for the creation of a colony. The highest scoring individual is selected as a queen. The queen is directly immigrated into the next generation and the remaining members of the generation are created from random mutations of the queen’s design string. This method can be seen visually in Figure 1.4.

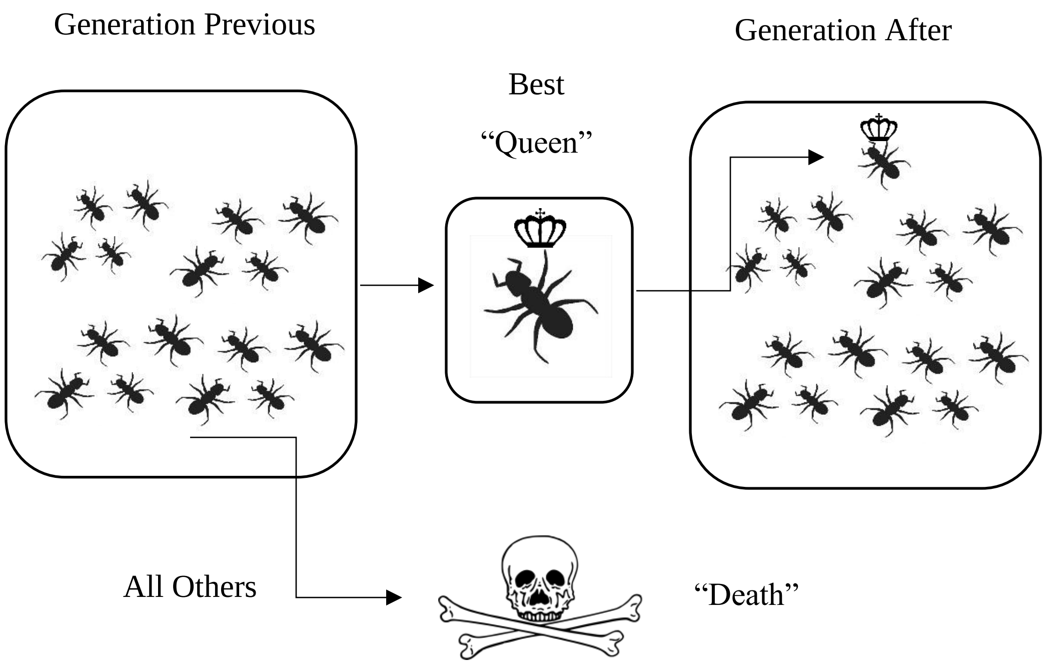


Figure 1.4 Killer Queen Method

The Roulette method shown in Figure 1.5 is dependent on statistical probability to ensure the continuance of a good design. Each designs objective function score is weighted against the sum of all objective function scores [13]. These weighted values represent a portion of a circle radiating from the center like a roulette wheel. The parents are selected at random by spinning the wheel twice. The resulting offspring then has some prescribed chance for mutations and crossover to occur. There is no elitism directly structured into the wheel spin. This produces a chance that a superior design may be neglected in future generations.

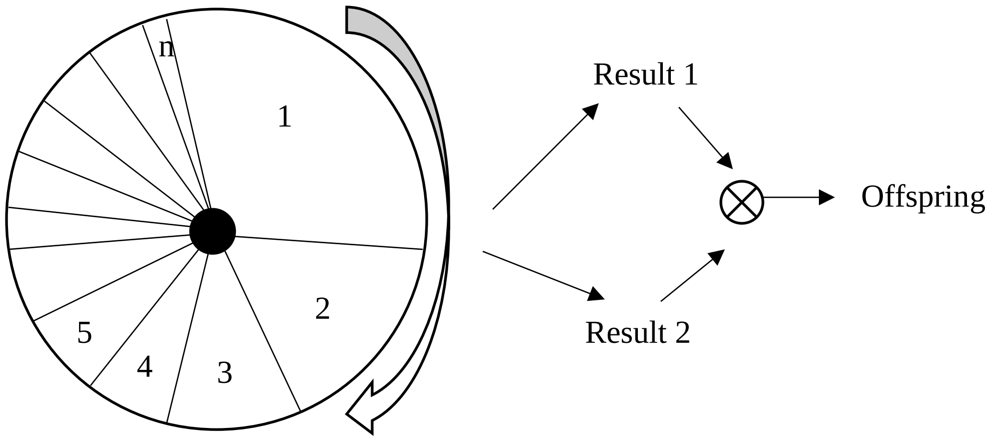


Figure 1.5 Roulette Method

The Tournament method incorporates a random selection process to select four individuals from the population. They are then made to “fight” against each other with the winner securing breeding rights and the loser returning to the population. The parents are the winners of both “bouts” and possess superior objective function scores to their competitors [13]. The random draw process continues until the next generation is full. A visual representation of this method can be seen in Figure 1.6.

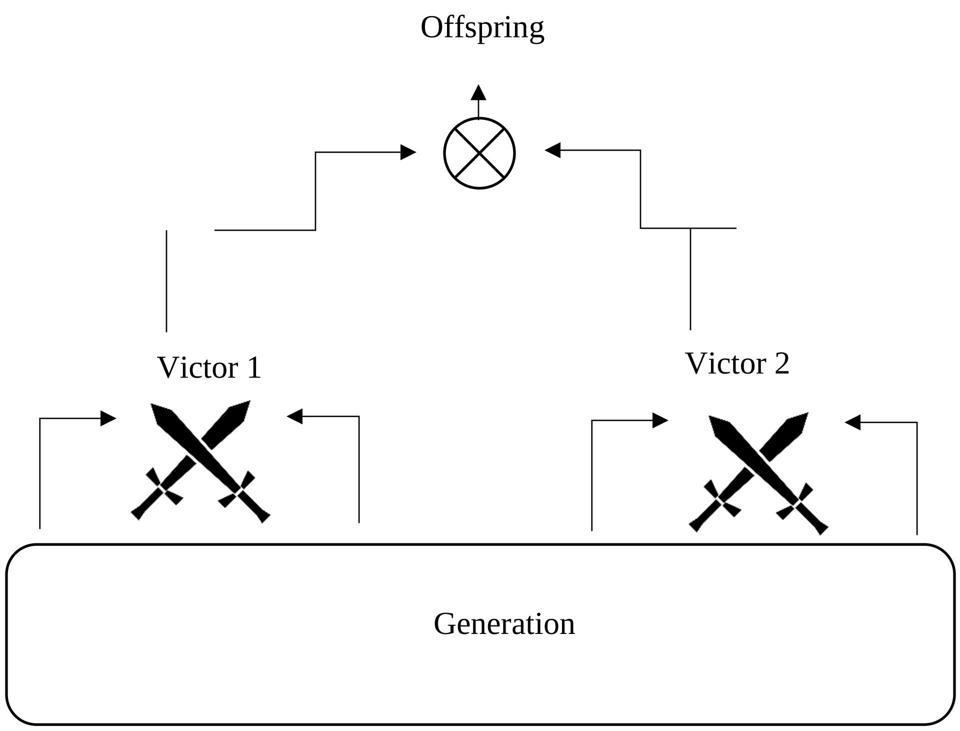


Figure 1.6 Tournament Method

The breeder pool method is a hybrid method created by Dan Raymer for use in genetic optimization routines of aircraft [13]. It combines elements of the three previous methods to provide a more stable path to the best aircraft. First the population is ranked, then a top percentage is removed and placed in a breeder pool. The rest of the population is discarded. The next generation draws only from the breeder pool which is automatically immigrated to the next generation. The additional levels cause the algorithm to converge more slowly than the killer queen optimization structure but the converged result is capable of surpassing the objective function score of the three previous optimization routines and will be the basis for the optimization routine developed herein. The breeder pool method can be seen in Figure 1.7.

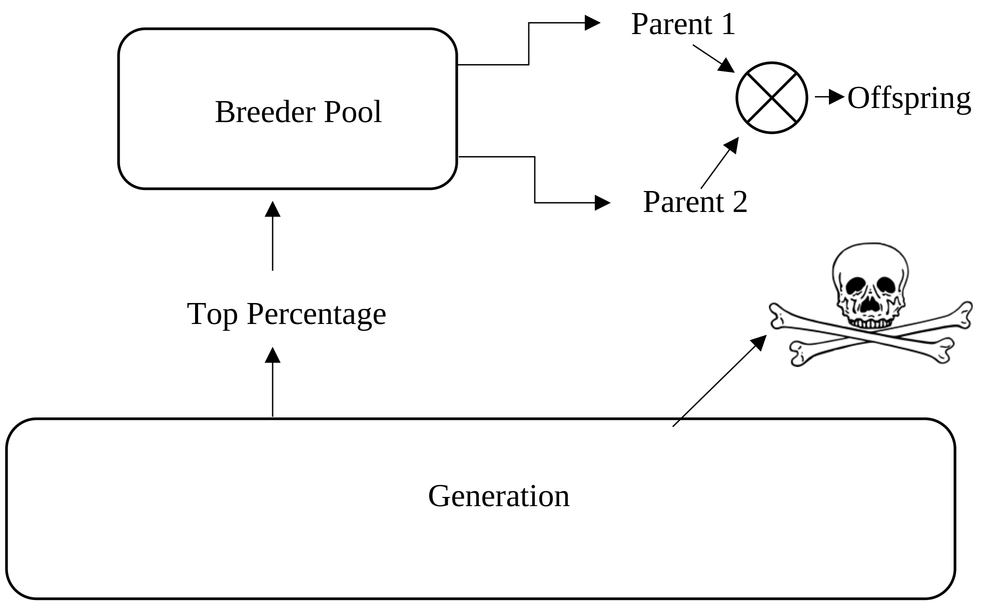


Figure 1.7 Breeder Pool Method

1.3 Problem Statement

1.3.1 Mission Based Genetic Algorithm

The Department of Defense budgets approximately 4.5 billion dollars a year in research and development and procurement of unmanned aerial vehicles. And it cites the time to field the latest technology as one of the improvement goals [21]. In order to provide the most efficient and cost effective UAV for a given mission, a genetic algorithm is proposed to augment the existing product development cycle shown in Figure 1.8. By utilizing the non-continuous design space, defined by a continuous and discrete variable design string, the algorithm will provide a thorough exploration of the design space in the preliminary design phase of aircraft design, the 0 to 40 percent section of the development cycle, with no need for a predetermined design layout. At its convergence, the resulting aircraft will represent a superior airframe suitable for continued development, reducing the overall cost of the research and design process and increasing the performance characteristics of the resulting aircraft in its requested mission profile.

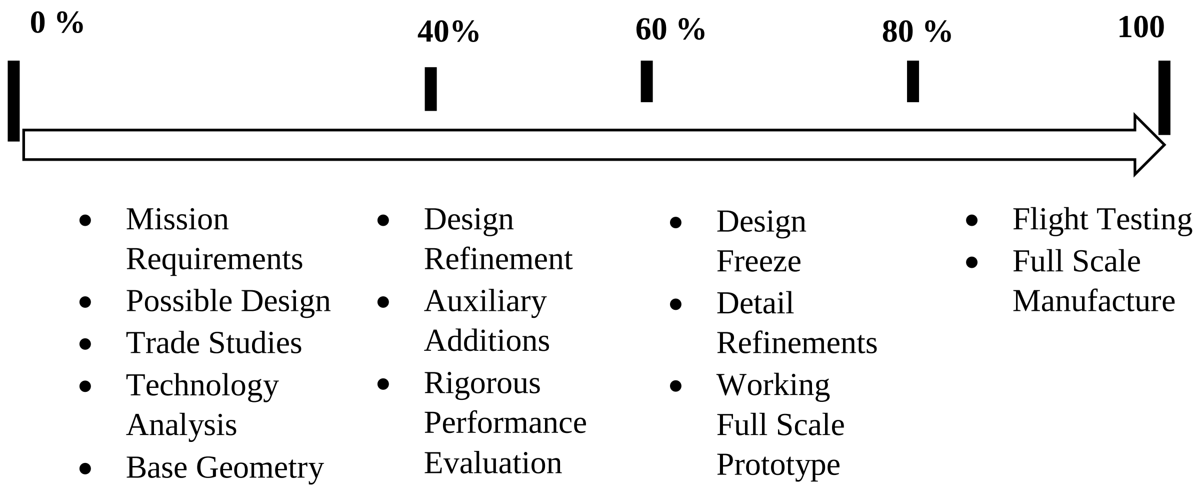


Figure 1.8 UAV Development Cycle

1.3.2 Phenotype Integration and the Tiered Design Space

In genetics phenotypes function as gene modifiers. They are small hydrocarbon chains that attach to individual genes. Phenotypes can turn genes on and off but are generally defined as how certain genes are represented [6]. In example, every human has an eye structure defined by a shared genome but iris color is a phenotype addition that changes how that structure is physically perceived. For aircraft tails, can be considered a shared genome but the type of tail can be defined as a phenotype addition that changes how that gene physically manifests.

By integrating discrete phenotype variables into the design string, the genetic algorithm proposed can weigh various choices of design features against each other. This inclusion permits the algorithm to explore the various combinations of base layout, tail configuration, engine type, and number of engines; while simultaneously evaluating wing, tail, fuselage, and engine geometry and performance.

CHAPTER 2

GENETIC ALGORITHM ARCHITECTURE

2.1 General Framework

The primary limitation of aircraft procurement programs is cost. Minimizing this cost while still providing the necessary performance aircraft provides the goal of the optimization algorithm. The general framework of this optimization routine is as follows:

Maximize: $f(\boldsymbol{x})$

Where: $f(\boldsymbol{x}) = f(x_1, x_2, x_3, \dots, x_n) = Performance - Cost \qquad i = 1\ to\ n$

Subject to: $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$

Where: $g_i(\boldsymbol{x}) \leq g_i(x_1, x_2, x_3, \dots, x_n) \qquad i = 1\ to\ m$

$h_i(\boldsymbol{x}) = h_i(x_1, x_2, x_3, \dots, x_n) \qquad i = 1\ to\ m$

Here $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$ represent two sets of constraints consisting of *m* equations. These constraints can be grouped into two main types, equality, and inequality constraints. The inequality constraints are the most prominent. These control the geometry limits, span, fuselage radius and length, tail size, and the mission requirements requested by the user, for example range, endurance, speed, etc. The equality constraints represent limitations on design layout corresponding to tail type, aircraft type, type of engine and number of engines. Additional inequality constraints can be added to require certain performance values.

2.2 Mixed Variable Design String

The design string is at the core of a genetic algorithm. It represents every possible combination of attributes, genes, of a proposed design condensed into as few design variables as possible. It is also referred to as a chromosome, as it performs the same function as the natural genetic structure in relaying the genetic information that defines the next generation. In order to fully encompass the design features of unmanned aerial vehicles the design string for the aircraft is chosen to have twenty genes. The individual genes their position in the design string, what the gene represents, and the type of variable associated with the gene are listed in Table 2.1.

Table 2.1 Genetic Chromosome for UAV

	Gene Number	Variable Name	Variable Type
	1	Aircraft Type	Discrete
Wing Geometry	2	Wing Half-Span	Continuous
	3	Wing Root Chord	Continuous
	4	Wing Dihedral	Continuous
	5	Wing Sweep (c/4)	Continuous
	6	Wing Taper	Continuous
Fuselage Geometry	7	Fuselage Diameter	Continuous
	8	Main Fuselage Length	Continuous
	9	Fuselage Nose Length	Continuous
	10	Fuselage Tail Length	Continuous
Tail Geometry	11	Tail Type	Discrete
	12	Horizontal Tail Span	Continuous
	13	Horizontal Tail Chord	Continuous
	14	Horizontal Tail Taper	Continuous
	15	Vertical Tail Span	Continuous
	16	Vertical Tail Chord	Continuous
	17	Vertical Tail Taper	Continuous
Propulsion System	18	Propulsion Type	Discrete
	19	Number of Engines	Discrete
	20	Propulsive Power/Force	Continuous

2.2.1 Aircraft Type

The aircraft type represented by the first gene in the design string, is a phenotype that controls the expression of the aircraft layout, where the tail and wing are located with respect to one another on the fuselage. The gene is represented as an integer from 1 to 3 with 1 identifying traditional layout, 2 representing a non-lifting canard layout, and 3 representing a flying wing see Figure 2.1.

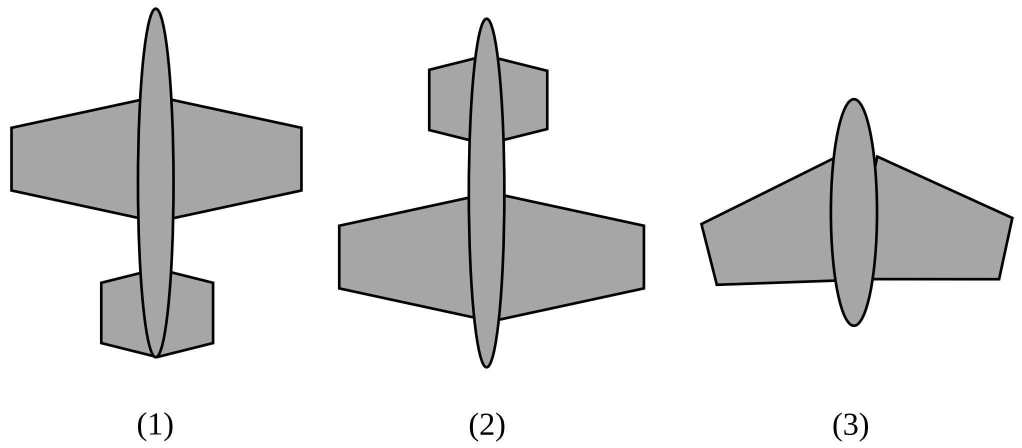


Figure 2.1 Aircraft Type

Aircraft type geometry layout (1) Traditional Aircraft Layout (2) Non-Lifting Canard (3) Flying Wing/Tailless

2.2.2 Wing Geometry

The wing represents arguably the single most important element of an aircraft. The geometry of the wing is the primary driver of performance metrics like range, endurance, and stability. To completely define the wing within the algorithm its geometry is decomposed into five independent variables, characterized by genes two thru six, with the genes representing the variables of half-span, root chord length, dihedral angle, quarter chord sweep angle, and taper ratio. This provides a basic wing shape that can be later refined by selection of airfoil and more complex geometry. The tip chord is defined as the taper ratio multiplied by the root chord. The geometry of the wing can be seen in Figure 2.2.

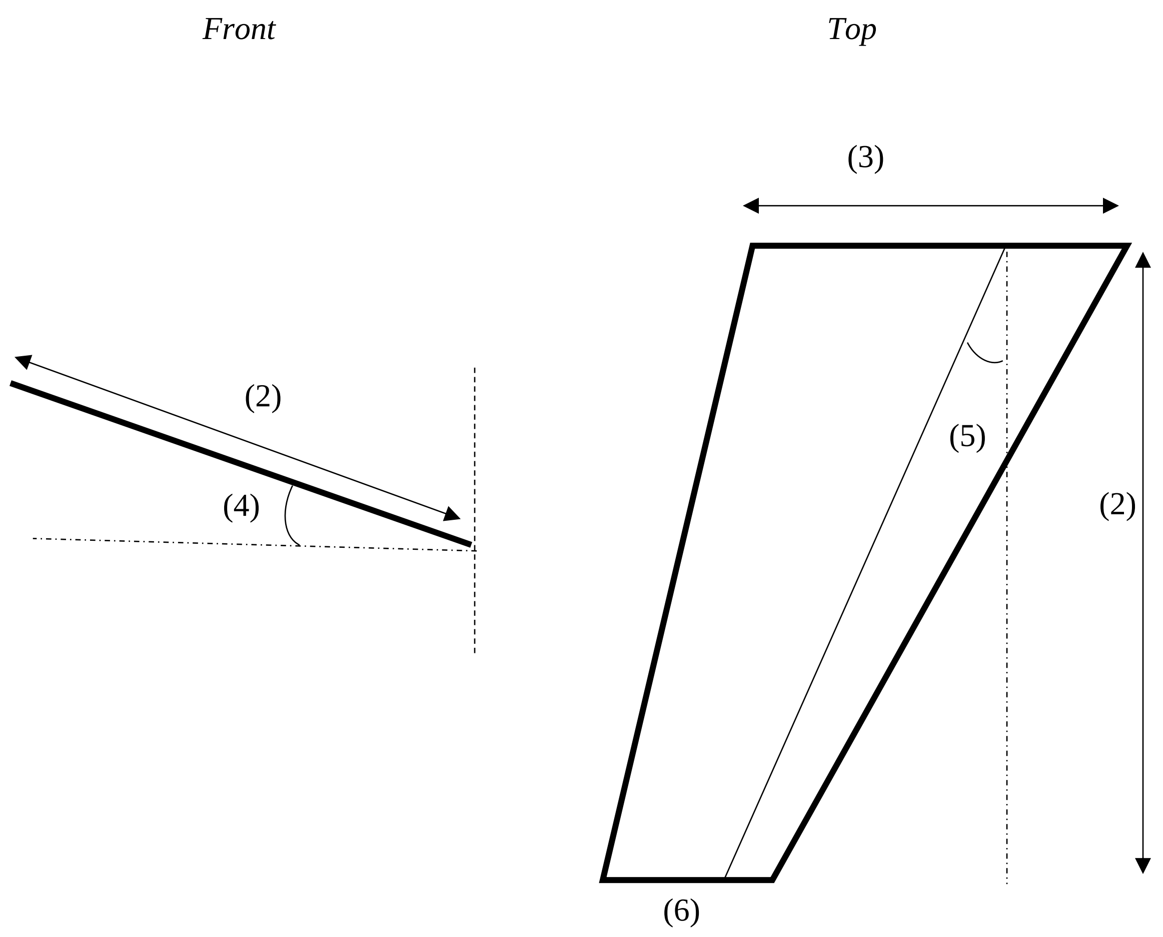


Figure 2.2 Half-Span Wing Geometry

Half Span Wing Geometry designated by the assigned gene number. Half-Span (2), Root Chord (3), Dihedral (4), Sweep (5), Taper Ratio (6). Notice Dihedral Angle (4) is exaggerated in the front view to better show its influence on the wing geometry. Taper ratio (6) in combination with the root chord (3) defines the tip chord, hence its location in the top view.

2.2.3 Fuselage Geometry

Four design variables define fuselage geometry. These variables are represented by genes seven through ten in the design string. The base level structure is defined as a cylinder of constant radius, gene seven, and length, gene eight. Two ellipsoid halves cap this cylinder. These halves begin with the same radius as the cylinder of the main body and each semi major axis is independently identified in the design string, gene nine for the nose partition, and gene ten for the tail partition.

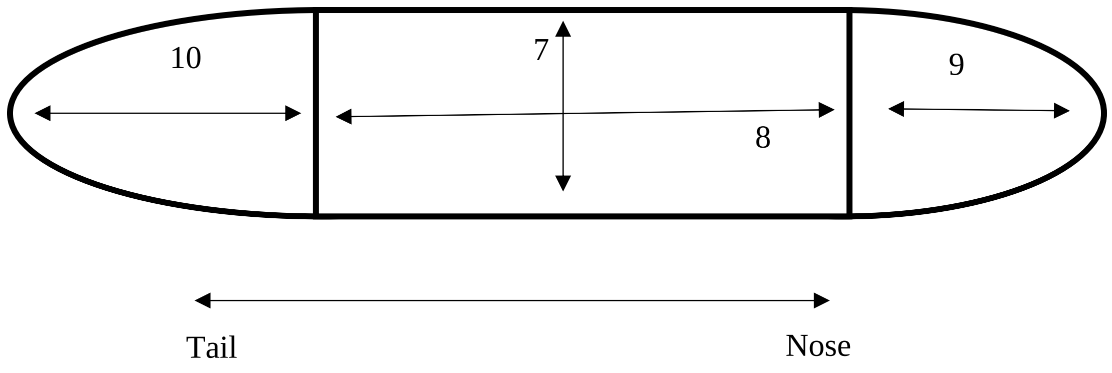


Figure 2.3 Aircraft Fuselage Side View

Aircraft Fuselage Side View Geometry as designated by the gene number in the design string: (7) Fuselage diameter, (8) Fuselage main body cylinder length, (9) Fuselage Nose Section Length, (10) Fuselage tail section length. The total length of the fuselage is then the sum of genes eight, nine, and ten.

2.2.4 Tail Geometry

Six continuous genes describe tail geometry, twelve thru seventeen, and the discrete expression of tail type, gene eleven. The algorithm is capable of assigning three distinct tail types, Traditional Style tails, T-Tails, V-Tails. Traditional tails are more common but there are instances where T-Tails and V-Tails can be useful, especially if fuselage length or stealth is a priority in the requested mission.

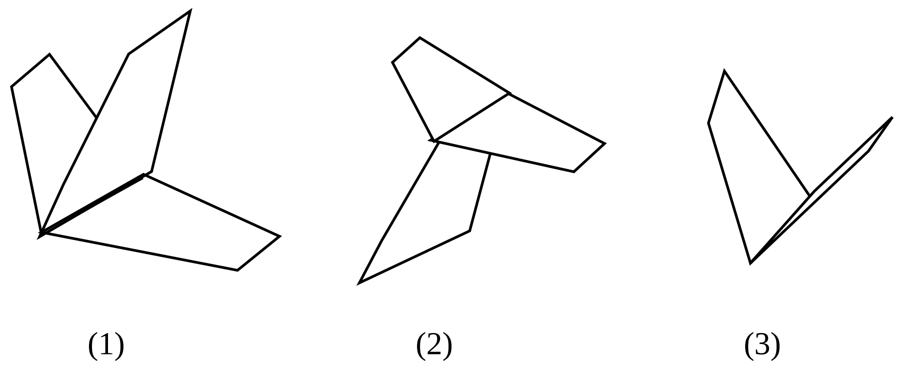


Figure 2.4 Tail Type Geometry

Tail Type Geometry possibilities within design space (1) Traditional Tail, (2) T-Tail, (3) V-Tail

The continuous variables then represent the dimensions of the tail structure. Genes twelve, thirteen, and fourteen describe the horizontal tail span, horizontal chord root and horizontal stabilizer taper ratio. Similarly, genes fifteen, sixteen, and seventeen represent the vertical stabilizer dimensions of span, root chord, and taper ratio respectively. The type of tail can define acceptable values of the other stabilizer; the T-Tail configurations horizontal stabilizer is dependent on the chord of the vertical stabilizer. The V-Tail is independent of the horizontal stabilizer genes and instead expresses its genes only for the vertical stabilizer just canted at some user selected angle usually forty-five degrees. All Horizontal tails have a sweep of thirty degrees and all vertical tails have a forty-five-degree sweep defined within the algorithm.

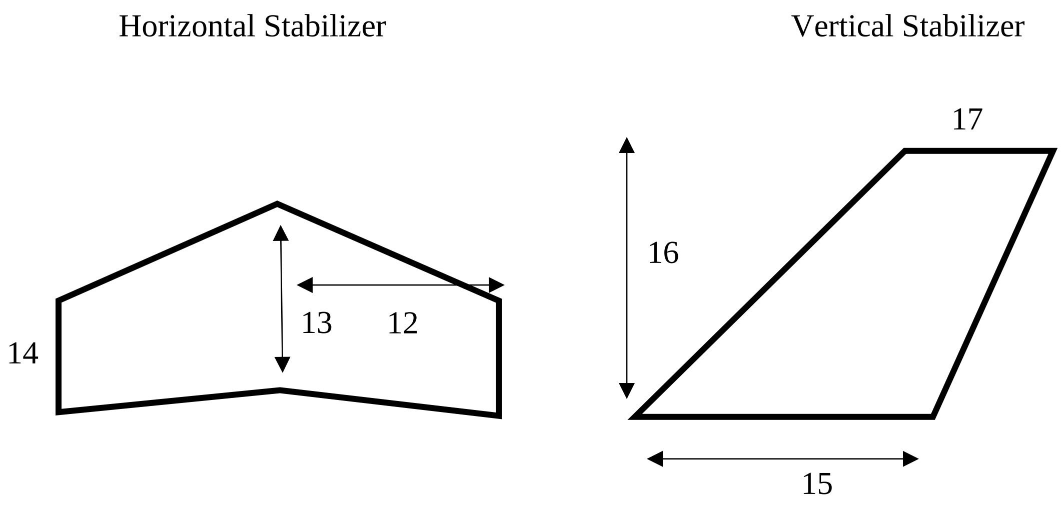


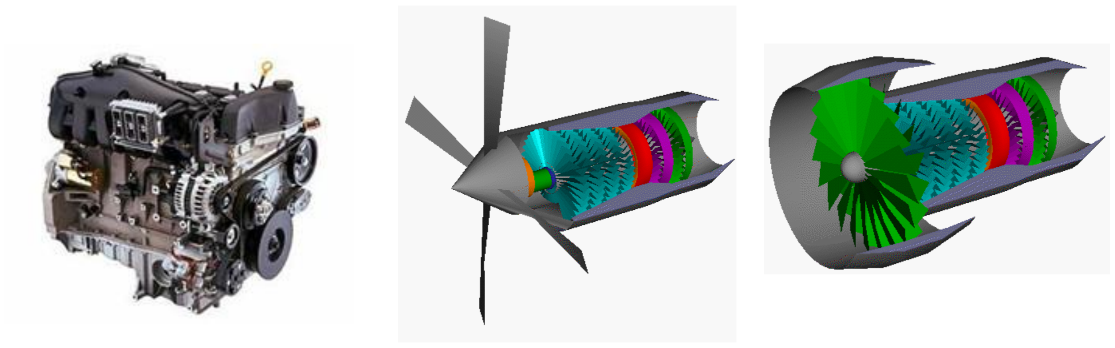
Figure 2.5 Tail Surface Geometry

Tail geometry with the position number of the gene defining it in the design string. Note that taper ratio 14 and 17 define the length of the tip chord of the tip section with respect to the root chord.

2.2.5 Propulsion System

There are three types of propulsion system widely seen in larger scale UAV design, considered for this algorithm. Electric propulsion which is becoming more and more common is not considered as its fuel consumption and power ratings do not align when directly compared to existing inline piston propeller engines, turboprop engines and low bypass turbofan engines. The number of engines is controlled by gene nineteen. UAVs of significant scale, Group 3 and above, rarely possess more than two engines. As such, designs with greater than two engines are not considered. Power per engine for piston and turboprop engines is assigned in gene twenty in the design string while the maximum thrust producible by the engine is assigned for jet turbofans.

For the purposes of this algorithm propulsion systems will be treated as custom to a particular airframe. This adds considerable cost when comparing the projected cost to the cost of utilizing stock engines, but permits near unlimited scaling of precision engines for a craft. The three engine types permitted within the algorithm are shown in Figure 2.6.



(1) (2) (3)

Figure 2.6 Engine Type Matrix

Propulsion elements listed by the corresponding integer in the design string. (1) Inline Piston engine, (2) Turboprop Engine, (3) Jet Turbofan Engine

2.3 Discrete Variable Design Space

With the inclusion of the discrete variables, the design string potentially identify aircraft whose design string represents a conflict. In example if gene one specifies a flying wing but then gene eleven specifies a T-Tail, which gene should be expressed in the design evaluation. To prevent this a hierarchy is imposed that limits the possible discrete variable combinations present in the design string. These constraints do not penalize the design in the evaluation phase and instead alter the interpretation of the design string. In Table 2.2, one can see the limited effect this has on the possible combinations of discrete elements with only the canard and flying wing type genes clashing with the tail type genes.

The need for the aircraft type to dominate is twofold. First it prevents aircraft configurations like tandem wing, three surface, or a flying wing with a tail. The aerodynamic analysis incorporated within this algorithm is incapable of accurately assessing these designs. Secondly its placement at the top of the design string allows aircraft types to be easily distinguished within the evaluation loop. These conflicting genes are shown visually in Table 2.2.

Table 2.2 Gene Interference

		Aircraft Type			**Tail Type**			**Engine Type**			**# Engines**	
		1	2	3	1	2	3	1	2	3	1	2
Aircraft Type	1	▨	▨	▨	X	X	X	X	X	X	X	X
	2	▨	▨	▨	X	O	O	X	X	X	X	X
	3	▨	▨	▨	X	O	O	X	X	X	X	X
Tail Type	1	X	X	X	▨	▨	▨	X	X	X	X	X
	2	X	O	O	▨	▨	▨	X	X	X	X	X
	3	X	O	O	▨	▨	▨	X	X	X	X	X
Engine Type	1	X	X	X	X	X	X	▨	▨	▨	X	X
	2	X	X	X	X	X	X	▨	▨	▨	X	X
	3	X	X	X	X	X	X	▨	▨	▨	X	X
# Engines	1	X	X	X	X	X	X	X	X	X	▨	▨
	2	X	X	X	X	X	X	X	X	X	▨	▨

2.4 The Cost Model

To approximate the cost of the procurement program and by extension the cost per aircraft associated with a specific design the DAPCA IV Model described in [12] is employed. The Development and Procurement Cost of Aircraft Model (DAPCA) was developed by the Rand Corporation and provides statistical approximations for the number of hours of labor from various departments and the cost of particular elements like engines, internal structures, material cost, machining elements, etc. The model described in detail below is intended to predict cost of a quantity Q of aircraft delivered over the course of a five-year program. The cost per hour of the general labor groups involved in the design process is displayed in Table 2.3.

Table 2.3 Departmental Labor Rates

Department	**Cost per Hour (USD 2012)**
Engineering (R_E)	115.00
Tooling (R_T)	118.00
Quality (R_Q)	108.00
Manufacturing (R_M)	98.00

The number of hours required for each department is then described by the equations

$$Engineering\ Hrs = \boldsymbol{H_E} = 5.18 * W_e^{0.777} * V^{0.894} * Q^{0.163} \tag{2.1}$$

$$Tooling\ Hrs = \boldsymbol{H_T} = 7.22 * W_e^{0.696} * V^{0.696} * Q^{0.263} \tag{2.2}$$

$$Manufacturing\ Hrs = \boldsymbol{H_M} = 10.5 * W_e^{0.82} * V^{0.484} * Q^{0.641} \tag{2.3}$$

$$Quality\ \ Hrs = \boldsymbol{H_Q} = 0.133 * H_m \qquad (2.4)$$

The cost per department is then approximated by equations 2.5 thru 2.7

$$Cost\ Engineering = C_E = \boldsymbol{H}_E * \boldsymbol{R}_E \qquad (2.5)$$

$$Cost\ Tooling = C_T = \boldsymbol{H}_T * \boldsymbol{R}_T \qquad (2.5)$$

$$Cost\ Manufacturing = C_M = \boldsymbol{H}_M * \boldsymbol{R}_M \qquad (2.6)$$

$$Cost\ Quality = C_Q = \boldsymbol{H}_Q * \boldsymbol{R}_Q \qquad (2.7)$$

Where W_e is empty weight, V is maximum velocity, Q is quantity of aircraft.

These costs are then added to development support costs, flight testing costs, with two testing aircraft, manufacturing materials cost, and engine cost.

$$Development\ Support\ Costs = \boldsymbol{C}_D = 67.4 * W_e^{0.630} * V^{1.3} \qquad (2.8)$$

$$Flight\ Test\ Costs = \boldsymbol{C}_{Flt} = 1947 * W_e^{0.325} * V^{0.822} * \#FTA^{1.21} \qquad (2.9)$$

$$Manufacturing\ Materails\ Cost = \boldsymbol{C}_{Mat} = 31.2 * W_e^{0.921} * V^{0.621} * Q^{0.799} \qquad (2.8)$$

$$Turbine\ Engine\ Production\ Cost = \boldsymbol{C}_{ENG} = 3112 * [9.66 * T_{max} + 243.25 * \dots M_{max} + 1.74 * T_{turbine\ inlet} - 2228] \qquad (2.9)$$

$$Piston\ Engine\ Production\ Cost = \boldsymbol{C}_{ENG} = 1200 * Power(kW) \qquad (2.10)$$

Where FTA is flight test aircraft, assumed to be two, T_{max} is the engine maximum thrust, and $T_{turbine\ inlet}$, is the operating temperature at the turbine stage, and M_{max} is the designed maximum Mach number for the engine. The cost of piston engines per unit power derived from [10].

The program cost is then:

$$Cost\ Program = C_E + C_T + C_M + C_Q + C_D + C_{Flt} + C_{Mat} + \#Engines * C_{ENG} \quad (2.11)$$

Avionics cost is then estimated as five percent of the total cost of the program leaving the total program cost as:

$$Total\ Cost = .05 * Cost\ Program + Cost\ Program \quad (2.12)$$

Continued adjustments can be made to the model to adjust for inflation and prices of materials. Commonly adjustment factors of 10 to 30 percent may be used to incorporate state of the art structural materials and other technologies into the predictive costs.

2.5 Mission Parameters

While cost often dictates the number of aircraft built and ultimately which design wins out, the airframes performance is just as important. The algorithm can produce several of the UAVs key performance metrics, including range, endurance, maximum velocity, takeoff distance, rate of climb, and radar cross section. The Table 2.4 lists all returnable performance values from the algorithm.

Table 2.4 Performance Metrics and Associated Variables

Performance Metric	Variable
Weight	W
Range	R
Endurance	E
Zero Lift Drag	CDo
Lift Curve Slope	CLalpha
Oswald Efficiency	e
Lift over Drag Ratio	LDmax
Maximum Lift Coefficient	CLmax
Maximum Velocity	Vmax
Cruise Velocity	Vcruise
Loiter Velocity	Vloiter
Rate of Climb (sea-level)	RCmax
Rate of Climb (altitude)	RCmaxalt
Stall Velocity (sea-level)	Vstall
Stall Velocity (altitude)	Vstallalt
Static Margin	StaticMargin
Takeoff Distance	takeoff
Landing Distance	landing
Maximum Turn Velocity	MaxTurnV
Maximum Turn Rate	MaxTurnRate
Maximum Turn Load Factor	nmax
Minimum Turn Radius	MinTurnRadius
Radar Cross Section	RCS

From these returnable values expectations on aircraft performance can be set. In example for a stealthy reconnaissance UAV that cruises at 30,000 feet that carries 1250 kg of payload and an additional 150 kg of auxiliary communications gear the following mission profile may be suggested:

Range Requested: 10000 km

Endurance Requested: 8 Hrs

Velocity Cruise Requested: 100 m/s

Static Margin: 15%

Radar Cross Section: 1 dB m^2

Alternatively, for a small tactical UAV operating from a short runway, carrying 50 kg of payload and an additional 12 kg in auxiliary, the following may be requested:

Range: 200 km

Endurance: 3 hours

Velocity Cruise: 30 m/s

Takeoff Distance: less than 1000 m

Rate of Climb: 5 m/s at sea-level

2.6 Constraining the Design Space

There are several constraints readily enforceable on the aircraft geometry without the need to specify which aircraft type. Table 2.5 lists the constraints on the design variables themselves. These hard constraints are penalized with a one billion-point penalty to the objective function. The severity of the constraint is necessary to prevent poor performing viable designs from being overlooked in favor of inviable designs within the selection process.

Table 2.5 Design String Constraints

Gene Number	Type	Minimum	Maximum
1	Discrete	1	3
2	Continuous	.5 meters	20 meters
3	Continuous	.5 meters	10 meters
4	Continuous	-7 degrees	7 degrees
5	Continuous	-30 degrees	30 degrees
6	Continuous	.1	1.0
7	Continuous	.2 meters	5 meters
8	Continuous	.2 meters	5 meters
9	Continuous	.2 meters	5 meters
10	Continuous	.2 meters	5 meters
11	Discrete	1	3
12	Continuous	0 meters	5 meters
13	Continuous	0 meters	5 meters
14	Continuous	.1	1.0
15	Continuous	0 meters	5 meters
16	Continuous	0 meters	5 meters
17	Continuous	.1	1.0
18	Discrete	1	3
19	Discrete	1	2

Notice the absence of gene number 20 in Table 2.5 is due to the statistical engine modeling incorporated into the algorithm. With three engine types and three separate statistical models each with their own empirical range of validity these constraints are enforced on a per aircraft basis.

2.7 Code Flow and Evaluation Diagram

The proposed algorithm utilizes generations of five-hundred aircraft represented by their twenty-variable design string to evaluate the cost and performance metrics associated with an unmanned aerial vehicle mission profile. The algorithm requires inputs regarding requested mission parameters importantly, requested operational altitude, payload weight, and auxiliary weight. The user can select then optional inputs to customize the geometry constraints such as needing to fit in a certain hangar, and accounting for runway limitations. The user may also choose to lock certain variables in position. This is particularly suitable for locking an aircraft configuration or engine type but may also be used to enforce predetermined design decisions. If for instance a designer knew they wanted a flying wing configuration regardless of the potential benefits of an alternative design the aircraft type gene, gene 1, could be locked as the flying wing phenotype three. The same can be said in any combination of discrete variables with the exceptions of the hierarchy limitations previously expressed in Table 2.2.

Once the user inputs are defined the algorithm proceeds through its subsequent sub functions evaluating design performance and then finally evaluating the objective function value for a full generation of designs. The actual genetic algorithm then takes the sorted generational data and performs a variant on the breeder pool algorithm developed by Dan Raymer [13]. This genetic algorithm driver alters the design strings and returns a second generation of aircraft. If after a set number of generations with no improvement the algorithm immigrates the leader and fills the remaining design strings with new aircraft with randomly generated design strings. This process of evaluation modification and elimination is then continued until a specified convergence criterion is reached. The overview of this optimization cycle can be seen in Figure 2.7.

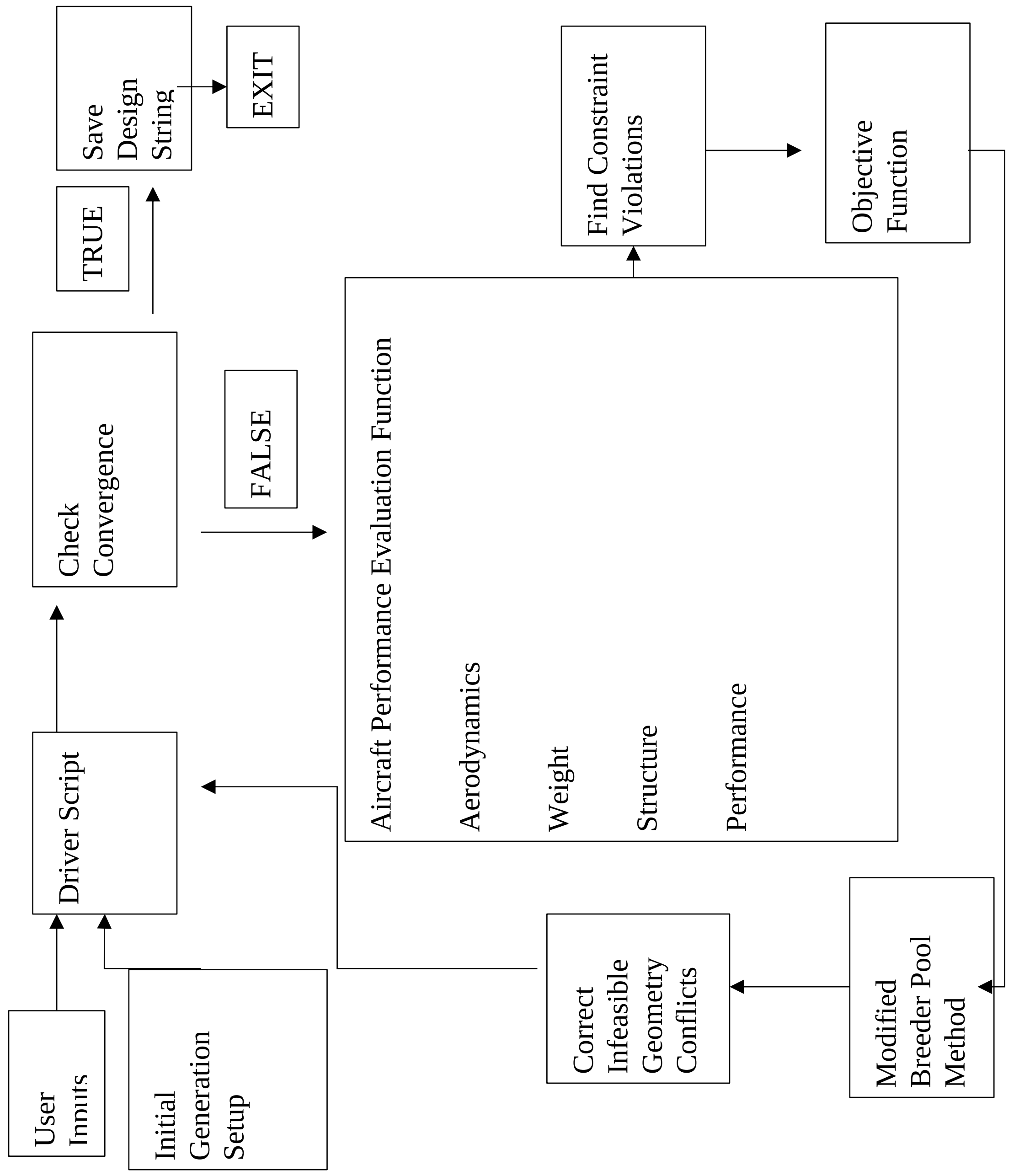


Figure 2.7 Code Flow Diagram

2.8 The Modified Breeder Pool Method

The main driver of the optimization cycle is the method that determines how when and where changes to the design string take place. Here a modified version of the original breeder pool method is utilized. The generation is sorted according to objective function score. The top twenty percent of the generation are placed into the breeder pool. The members of this pool are automatically immigrated into the new generation. The remaining eighty percent of the old generation are discarded. The next sixty percent of the new generation are created by randomly selecting two members of the breeder pool and performing a crossover operation. After the crossover operation, there is a chance for mutations to occur on the new offspring's design string. The mutations have a minimum modification of plus or minus five percent of the original design string value. For the discrete variables, any mutation results in a modification of plus or minus one. The remaining twenty percent of the new generation are then randomly generated by the same start function that began the optimization routine. If a design has remained the "leader", or best design, for longer than a set number of iterations, in this case one thousand, then a new generation is instead created by discarding all the designs except for the leader, in what can be dubbed an "extinction event". The remaining members of the new generation are randomly created using the starting design string generator. A pictorial representation of this architecture is shown in Figure 2.8.

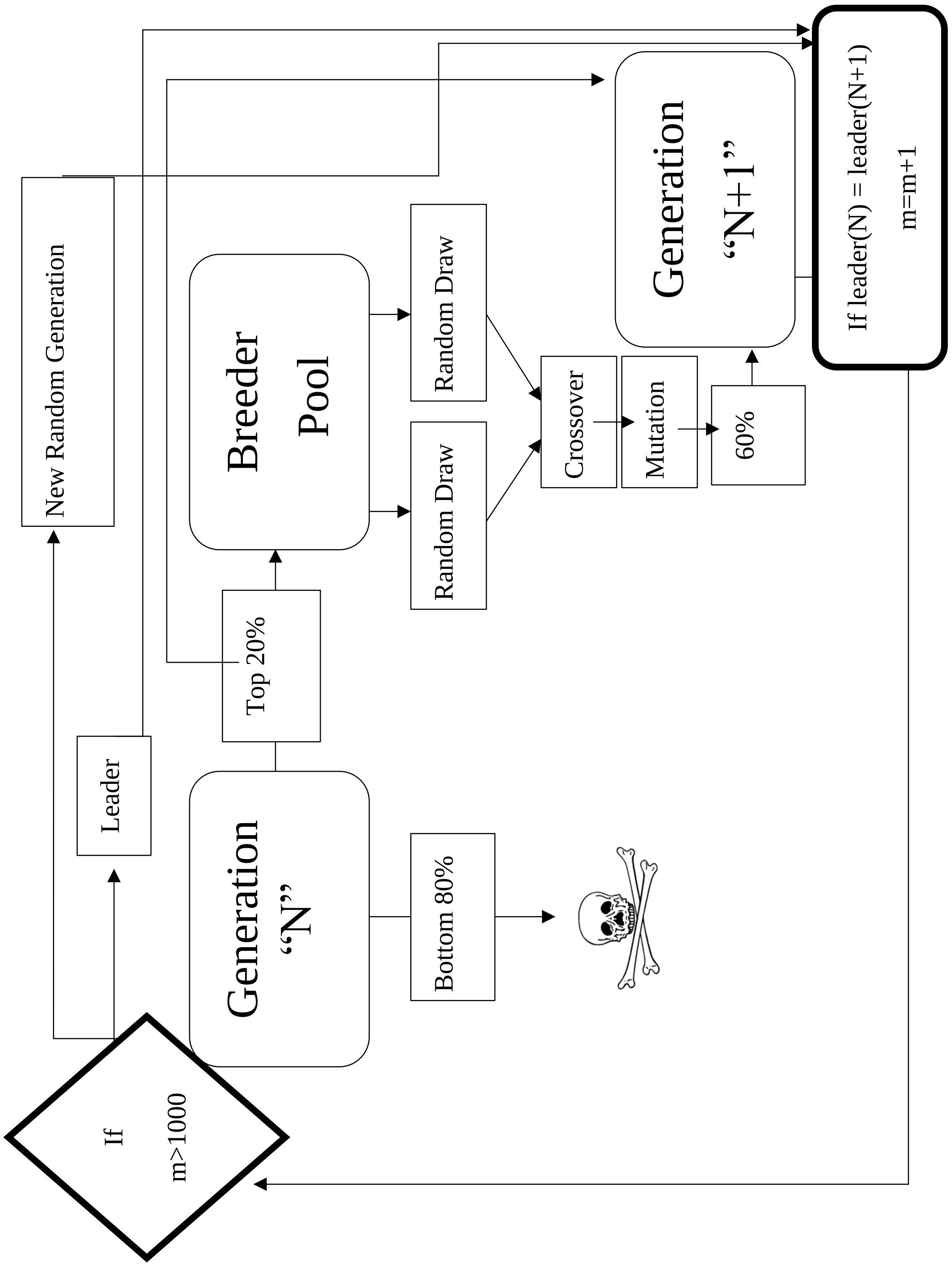


Figure 2.8 Modified Breeder Pool Method

2.9 Statistical Modeling and Methods of Performance Evaluation

The large design space available to the algorithm as well as the differences in evaluating particular elements like different tails and different engine types using as simple of a design string as possible necessitate empirical models for the remaining elements of the geometry or performance not directly specified in the design string.

2.9.1 Aircraft Weight Estimation

Aircraft weight effects a great deal of performance characteristics. In order to explore as much of the design space as possible weight was not expressly described within the design string. To estimate the weight of the airframes a weight build up model from [12].

The weight of the aircraft is approximated by a weight per unit surface area estimation multiplied by the surface area or wetted area of the existing part. A fraction of the total weight then is used to estimate landing gear weight.

Aircraft Weight Estimation Model

$$Weight\ of\ Wing = W_W = \boldsymbol{S}_{exposed} * 20\, {}^{kg}/_{m^2} \tag{2.13}$$

$$Weight\ of\ Fuselage = W_f = \boldsymbol{S}_{wet} * 15\, {}^{kg}/_{m^2} \tag{2.14}$$

$$Weight\ of\ Horizontal\ Tail = W_{Horz} = \boldsymbol{S}_{exposed} * 12\ {}^{kg}/_{m^2} \tag{2.15}$$

$$Weight\ of\ Vertical\ Tail = W_{Vert} = \boldsymbol{S}_{exposed} * 12\ {}^{kg}/_{m^2} \tag{2.16}$$

$$Weight\ of\ Installed\ Engines = W_{Enginst} = \boldsymbol{W}_{engine} * \boldsymbol{N}_{Engines} * 1.3 \tag{2.17}$$

$$Weight\ of\ Fuel = W_{Fuel} = \rho_{Fuel} * Volume_{Fuel} \tag{2.18}$$

Where volume of fuel is estimated as one third of the wing volume between the forward and rear spars with an additional one half of the fuselage volume.

$$Weight\ of\ Landing\ Gear = W_{Gear} = \boldsymbol{W}_{TOGW} * 0.033 \tag{2.19}$$

$$Weight\ Total = W_W + W_{Horz} + W_{Vert} + W_{Fuse} + W_{Enginst} + W_{Fuel} + W_{Gear} + W_{Payload} + W_{Auxiliary} \tag{2.20}$$

2.9.2 Engine Weight and Sizing

The multiple types of engines possible within the design space require individual models for each type. These models come with empirical limitations that are then enforced on the design space as hard constraints.

Inline Piston Engine Model

Empirical Power Range 75 kW to 225 kW

$$Weight\ of Engine = W_{Engine} = 2.98 * Power(kW)^{0.780} \tag{2.21}$$

$$Length\ of Engine = L_{Engine} = .17 * Power(kW)^{0.424} \tag{2.22}$$

$$Diameter\ of\ Engine = D_{Engine} = 0.5 \tag{2.23}$$

$$Height = H_{Engine} = 0.5 \tag{2.24}$$

$$Specific\ Fuel\ Consumption = SFC = \frac{0.068\ mg}{W-s} \tag{2.25}$$

$$Blade\ Diameter(3 - Blade) = 0.52 * Power(kW)^{0.25} \tag{2.26}$$

Where blade diameter is the propeller disk size for a 3-bladed propeller.

Turboprop Engine Model

Empirical Power Range 370 kW to 3600 kW

$$Weight\ of Engine = W_{Engine} = 0.96 * Power(kW)^{0.803} \tag{2.27}$$

$$Length\ of Engine = L_{Engine} = .12 * Power(kW)^{0.373} \tag{2.28}$$

$$Diameter\ of\ Engine = D_{Engine} = .25 * Power(kW)^{0.120} \tag{2.29}$$

$$Height\ of\ Engine = Diameter \tag{2.30}$$

$$Specific\ Fuel\ Consumption = SFC = 0.85 \frac{mg}{W-s} \tag{2.31}$$

Again equation 2.26 is used for the diameter of the propeller

$$Blade\ Diameter(3 - Blade) = 0.52 * Power(kW)^{0.25}$$

Turbofan Engine Model (Low Bypass Ratio Only)

This model is intended for engines with a thrust range of 15 kN to 300 kN and all turbofans within this model are assumed to have a bypass ratio, BPR, of two.

$$Weight\ of Engine = W_{Engine} = 14.7 * Thrust^{1.1} * e^{-0.045*BPR} \tag{2.32}$$

$$Length\ of Engine = L_{Engine} = 0.49 * Thrust^{0.4} * 0.9^{2} \tag{2.33}$$

$$Diameter\ of\ Engine = D_{Engine} = 0.15 * Thrust^{0.5} * e^{(0.04*BPR)} \tag{2.34}$$

$$Specific\ Fuel\ Consumption = SFC = \frac{22.7\ mg}{Ns} \tag{2.35}$$

$$Height\ of\ Engine = Diameter \tag{2.36}$$

2.9.3 Aerodynamic Modeling

The drag force on the aircraft is modeled using the drag buildup method described in [12]. Each component of the aircraft is evaluated for drag and then the total is summed together to provide the total zero lift drag coefficient for the craft in the subsonic regime.

$$C_{D_O} = \frac{\sum(C_{f_i} * FF_i * Q_i * S_{wet_i})}{S_{ref}} + C_{D_{misc}} + C_{D_{L\&P}} \tag{2.37}$$

Where *i* represents values due to individual components, C_f is the coefficient of friction, FF is form factor, Q is interference factor and S_{wet} is wetted area. The miscellaneous drag and leakage and protuberance drag are then assumed to be five percent of the total drag.

The algorithm assumes completely turbulent flow over all surfaces leading to the following equation for coefficient of friction.

$$C_f = \frac{0.455}{(\log(R))^{2.58} * (1+0.144 * M^2)^{0.65}} \tag{2.38}$$

Where R is Reynolds number and M is Mach number.

For the wing and tail sections form factor can be calculated from the equation:

$$FF = \left[1 + \frac{0.6}{({}^{x}/_{c})m} * \left(\frac{t}{c}\right) + 100 * \left(\frac{t}{c}\right)^4\right] * [1.34 * M^{0.18} * \cos(\Lambda)^{0.28}] \tag{2.39}$$

Where t/c is the thickness to chord ratio, x/c is the point of maximum thickness of the airfoil, and Λ is the sweep of the wing section.

For the Fuselage from factor is:

$$FF = \left(1 + \left(\frac{60}{f^3}\right) + \left(\frac{f}{400}\right)\right) \tag{2.40}$$

Where

$$f = \frac{l}{d} \tag{2.41}$$

Where *l* is fuselage length and *d* is fuselage diameter.

The lift slope is an important measure of aircraft aerodynamic performance, giving the coefficient of lift for an angle of attack. The equation 2.42 is the DATCOM method for estimating the lift curve slope of a wing versus angle of attack [12].

$$C_{L_\alpha} = \frac{2*\pi*AR}{2*\sqrt{\left(4+\frac{AR^2-1-M^2}{\eta^2}\right)*(1+\tan(\Lambda)^2)}} * \left(\frac{S_{exp}}{S}\right) * F \quad (2.42)$$

Where η is the Mach Correlation Airfoil Efficiency set at 0.95 for use in this routine.

Oswald efficiency gives a measure of how closely the lift distribution of a given wing compares with that of an elliptical distribution or its span efficiency. The higher this efficiency the less impact lift induced drag has on the aircraft. There are many methods for estimating Oswald efficiency [11]. Evaluating Oswald efficiency at early preliminary stage requires the use of multiple methods.

Oswald efficiency is calculated in two stages within this algorithm. The first calculates the wings Oswald efficiency by incorporating a leading-edge suction technique and then using an interpolation method from the tables as described by Samoylovitch [14] and returns the Oswald efficiency. Then using the nonplanar adjustment described in [11] adjusts the efficiency to account for the nonplanar dihedral effect. The Oswald efficiency at zero leading edge suction is dependent on Aspect Ratio, AR the compressibility correction β and the lift curve slope C_{L_α}.

$$e_{w/S_{e=0}} = C_{L_\alpha}/(\beta * \pi * AR) \quad (2.43)$$

The Oswald efficiency at perfect leading edge suction

$$e_{w/S_{e=1}} = \frac{C_{L_\alpha}}{\beta * AR} * \tilde{y}_{cg}^{e} \tag{2.44}$$

Adds a new term $\tilde{y}_{cg}^{e}$ which is the distance between the trailing vortex centers of gravity in the Treffetz plane.

The leading-edge suction of the actual craft is then

$$S_e = 0.974 - 0.0976 * e^{-0.456\left(AR*\frac{\lambda}{\cos(\Lambda)}\right)} \tag{2.45}$$

For craft with modest leading edge curvature. The Oswald efficiency, **e**, is then

$$e = e_w * k_f \tag{2.46}$$

Where k_f is the correction factor that incorporates the influence of the fuselage cross section. This factor is interpolated from the data presented in [14] and e_w is:

$$e_w = \frac{e_{w/S_e=1} * e_{w/S_e=0}}{S_e e_{w/S_e=0} - (1 - S_e) * e_{w/S_e=1}} \tag{2.47}$$

Finally, the Oswald efficiency is adjusted for nonplanar effects [11]

$$e_\Gamma = e\left(\frac{1}{\cos(\Gamma)}\right)^2 \tag{2.48}$$

Where Γ is the dihedral angle.

2.9.4 Maximum Velocity

The maximum velocity achievable by an aircraft is a strong measure of its performance. The following equations can be found in [4]. For a jet turbofan engine, the maximum velocity is relatively easy to calculate. It is found when the acceleration of the aircraft reaches zero in other words when maximum thrust is equivalent to drag. The thrust available at altitude for a turbofan is given in equation

$$Thrust\ Available = T_A = T_{sealevel} * \left(\frac{\rho_{alt}}{\rho_{sealevel}}\right)^{0.6} \tag{2.49}$$

The maximum velocity is then found from equation

$$V_{max} = \left(\frac{\frac{T_A}{W}*\frac{W}{S}+\frac{W}{S}*\sqrt{\left[\frac{T_A}{W}\right]^2-4*C_{D0}*K}}{\rho*C_{D0}}\right)^2 \tag{2.50}$$

Where

$$K = \frac{1}{\pi*e*AR} \tag{2.51}$$

The same cannot be said for propeller driven craft as their propulsion units are measured more typically in power. To accommodate all three types a power relation must be solved for the maximum velocity. Equations and give the power available at altitude for both reciprocating and turboprop engines.

$$Power\ Available = P_A = P_{sealevel}\left(1.132\left(\frac{\rho_{alt}}{\rho_{sealevel}}\right) - 0.132\right) \tag{2.52}$$

$$Power\ Available = P_A = P_{sealevel}\left(\frac{\rho_{alt}}{\rho_{sealevel}}\right)^{0.7} \tag{2.53}$$

The velocity must then be solved for using the relation

$$P_A = \frac{1}{2}\rho V^3 S C_{D0} + \frac{W^2}{\frac{1}{2}\rho V S} * K \tag{2.54}$$

To accomplish this an iterative evaluation process is completed within the algorithm trying velocities from zero to the speed of sound in steps of one-half a meter per second. The velocity that matches the available power is then returned to the main evaluation algorithm. The velocity of the aircraft is then limited to the speed where its previously sized aircraft does not break the speed of sound in full throttle flight.

The remaining performance variables have all been well documented in [1][4][12]. The remaining equations used in the performance evaluation phase of the algorithm can be found in Appendix A.

2.10 Convergence

The convergence criterion for this algorithm, when the algorithm terminates its search of the design space, is defined as ten thousand generations without any improvement in the current best design, simultaneously there must be no violation of any constraint, else the genetic algorithm continues. Alternatively, due to the size of the design space, the algorithm will terminate if it exceeds a set number of generations. The initial selected mark was chosen to be a quarter of a million generations. If by this time a dominant design is not found the program terminates and returns the current leader.

CHAPTER 3

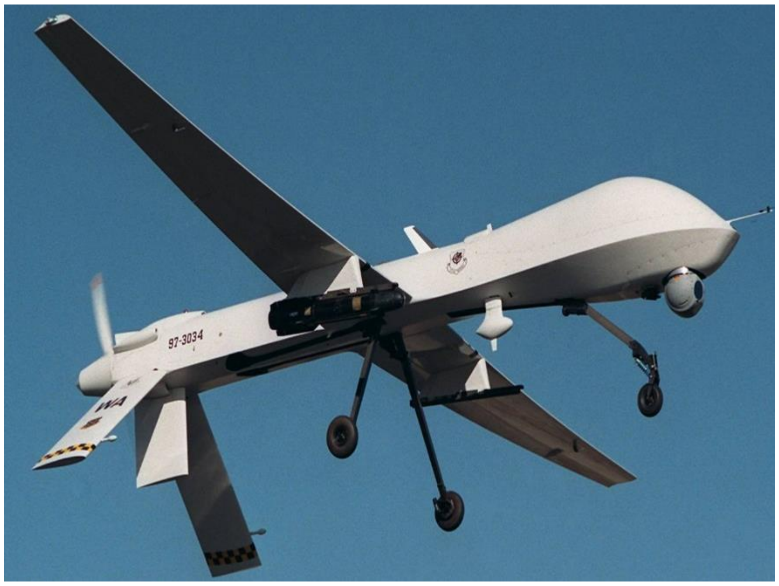
ALGORITHM AND AIRCRAFT BENCHMARKING

3.1 Performance Evaluation

To confirm the reliability of the evaluations of the performance variables within the algorithm design strings were assembled based on information of currently operation UAVs. The results of the performance output from the evaluation loop were then compared to the existing performance specifications. The evaluation loop was tested against the approximated design strings of the MQ-1 Predator, MQ-9 Reaper, and RQ-4 Global Hawk UAVs based on the information from the United States Air Force Aircraft Data Sheets [18] [19] [20].

3.1.1 MQ-1 Predator UAV Benchmark

The MQ-1 Preadtor is a low altitude reconnasiance and strike aircraft currently in use by the United States armed forces. The Predator is on the lower end of Group 4 UAV only crossing the weight barrier when fully loaded. It is powered by a single reciprocating four-cylinder engine. It has relatively limited range when compared to larger Group 4 UAVs [18]. The Predator and its design string are shown in Figure 3.1



Predator Design String
1
8.4 (m)
2 (m)
0
5 (deg)
0.25
0.4 (m)
2.74 (m)
2.74 (m)
2.74 (m)
3
0
0
0
2.1(m)
0.4 (m)
0.5
1
1
85 (kW)

Figure 3.1 Predator UAV and Design String

The Predator's actual performance specifications compared to the evalutaion routines performance is shown in Table 3.1

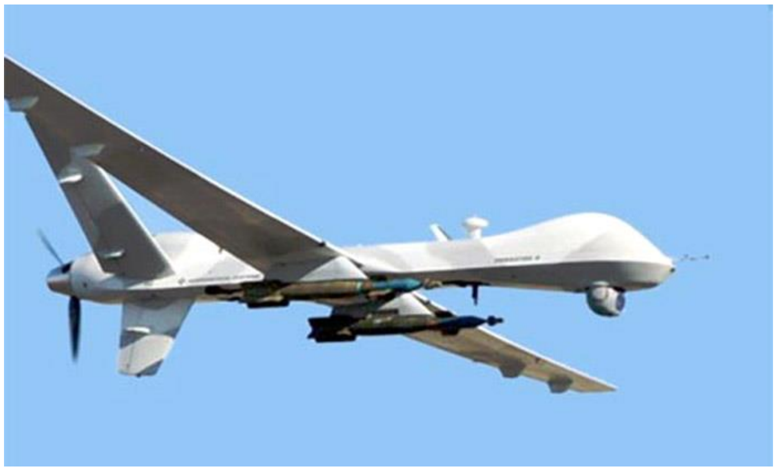
Table 3.1 Predator UAV Performance Comparison

	Weight Fuel (kg)	Weight (kg)	Range (km)	Velocity Cruise (m/s)	Velocity Max (m/s)	Cost per Aircraft (1e6 USD)
Calculated	201	1073.1	2159	44.0	57.5	17.9
Actual	300	1020	1240	37.5	60	5.0
% Difference	33.3	5.21	74.1	17.3	4.16	258

Given the approximations necessary to fully decompose the complete geometry of the Predator the errors seen in table are not as severe as may be interpreted on first inspection. The weight of fuel within the algorithm is severely limited within the geometry of the aircraft. No aircraft for example may have more than two thirds of its total weight be fuel. For range estimation the application of a constant fuel consmption regardless of airspeed and ignoring the decreases in efficiencies in propeller and shaft power transmission can easily explain the almost doubled range. Total weight and velocity provide good comparisons to the existing airframe. This suggests that the drag model incorporated is reasonably accurate as is the power available curve solver for reciprocating engines. The real concern is cost, Predators are sold in serialized batches of four aircraft meaning the unit cost of 20 million USD is much more comparable to the predicted cost per aircraft returned by the evaluation aircraft.

3.1.2 MQ-9 Reaper UAV Benchmark

The MQ-9 Reaper is a turboprop powered attack platform. It carries triple the payload of the Predator, hauling 1700 kg of weaponry and equipment. The V-Tail design supports a large aspect ratio wing that carries the majority of the aircrafts fuel and also supports weapons hard points [19]. The Reaper and its approximated design string can be seen in Figure 3.2.



Reaper Approximated Design String	
	1
	10.05 (m)
	2 (m)
	0
	0 (deg)
	0.8
	0.6 (m)
	3.66 (m)
	3.66 (m)
	3.66 (m)
	3
	0
	0
	0
	5.0 (m)
	1.5 (m)
	0.35
	2
	1
	671 (kW)

Figure 3.2 Reaper UAV and Design String

The Reapers's actual performance specifications compared to the evaluation routines predicted performance is shown in Table 3.2

Table 3.2 Reaper Performance Comparison

	Weight Fuel (kg)	Weight (kg)	Range (km)	Velocity Cruise (m/s)	Cost per Aircraft (1e6 USD)
Calculated	879	4621	2322	97.5	46.35
Actual	1814	4760	1852	103	16.05
% Difference	51.5	2.9	25.3	5.339	189

Again, the errors in the approximate methods of reducing the Reaper's geometry are evident, primarily in fuel capacity and in range. The locking of fuel consumption to be a single constant rate regardless of the flight conditions being evaluated only exasperates this effect. The best match again is total weight which despite the fuel discrepancy is fairly accurate. Cost continues to be greatly over estimated, though less so than the Predator evaluation. Again, the addition of custom engine design and development in the process instead of a stock engine and the limits of forty aircraft delivered over five years contribute to this high cost discrepancy.

3.1.3 RQ-4 Global Hawk UAV Benchmark

The RQ-4 Global Hawk is a Group 5, long range, high endurance, reconnaissance aircraft currently in use by the United States military and NASA. The V-Tail conventional design is powered by a single jet turbofan and operates at a much higher altitude than the Reaper and Predator. The wingspan sits at just under forty meters and represents the upper span limit incorporated into the algorithms constraints. The Global Hawk currently holds the record for longest unrefueled flight by a United States Air Force vehicle at 34.3 hours [20]. The Global Hawk and its approximated design string can be found in Figure 3.3.

Reaper Approximated Design String	1
	19.9 (m)
	3 (m)
	0
	15 (deg)
	0.3
	1.8 (m)
	5.0 (m)
	4.0 (m)
	4.0 (m)
	3
	0
	0
	0
	4.0 (m)
	1.5 (m)
	0.35
	3
	1
	36 (kN)

Figure 3.3 Global Hawk UAVand Design String

The Global Hawk's evaluated performance compared with its listed performance can be found in Table 3.3.

Table 3.3 Global Hawk Performance Comparison

	Weight Fuel (kg)	Weight (kg)	Range (km)	Endurance (hours)	Velocity Max (m/s)	Cost per Aircraft (1e6 USD)
Calculated	6746	13846	19160	8.6	166.7	161.1
Actual[16]	6781	14628	22780	34+	160	104*
% Difference	0.52	5.3	15.9	74.0	4.19	35.4

*Base model estimation only

The design string much more closely approximates the approximated Global Hawk as evidenced by the errors in weight of fuel, weight, and maximum velocity. Cost error is much less exaggerated likely due to the size of the aircraft being more similar to what the DAPCA IV model was intended to evaluate, something on the order of a Boeing 737 commercial liner as opposed to a Cessna 172. The range and endurance divergence is not disqualifying for a simple reason. In the model the aerodynamics were assumed to be that of a flat plate not an airfoil. The true lift to drag ratio of the Global Hawk is estimated around 36 but the model predicts that a similarly sized UAV can only have a max lift over drag ratio of 11. This difference is the main driver for under estimation of both range and endurance.

3.2 Design Space Oddities

During the optimization routine, the design space is explored very rapidly. This speed coupled with the large size of the design space as a whole allows for some designs to become either dominant or just represent an interesting point in the design space. One of these cases the “Death Star” case, never makes an impact on the actual finished result but it can randomly appear at the beginning of the routine or after one of the extinction event random repopulations. The second case the “Imploding Star” was discovered by accident by allowing design in an unconstrained design space. This case exemplifies the effect fuselage diameter holds over the objective function of cost only.

3.2.1 The Death Star Case

During the random design string setup, the fuselage occasionally becomes so large that the lifting surfaces become completely enclosed within the fuselage. The resulting body often has two huge engines in the upper echelons of the design space. The design string can be seen in Table 3.4.

Table 3.4 The Death Star Design String

The Death Star Design String																			
3	2	1.25	0	0	.2	5	.1	5	5	1	0	0	0	0	0	0	3	2	300

Despite its appearance as a feasible design, the performance requirements find it desperately lacking. It weighs much more than a typical aircraft of similar scale in part due to large engines but mostly due to the enormous volume available for fuel. Perhaps even more disqualifying is its inability to generate lift with no exposed wing surface area, taken with a coefficient of drag about ten and a negative rate of climb at sea level this design string represents an almost comical solution. However, it does appear throughout the optimization routine when random design strings are generated.

3.2.2 The Imploding Star Case

In early versions of the optimization routine no constraint was placed on the low end of design variables. Since cost is the primary objective function rather than minimize size and material needs of the aircraft as a whole, a single variable, fuselage radius, gene seven, which directly controls the cost associated with all three sections of the fuselage, diverged towards negative infinity causing the cost to also diverge towards negative infinity. Thus, the design string in Table 3.5 became the leader and continued to propagate forward.

Table 3.5 The Imploding Star Design String

Imploding Star Design String																			
1	a	b	c	d	e	-1e38	f	g	h	1	i	j	k	l	m	o	1	1	p

The lettered genes represent values that due to the extreme value of the diameter the value is unable to resolve in the limited format in MATLAB. While this case is not possible within the fully constrained framework presented by this research, it is worth noting the power of optimization routines to exploit even the smallest weakness in the coded limitations. In this case the routine successfully located and abused the diameter of the fuselage in regards to surface area to completely undermine the utility of the cost function.

CHAPTER 4

AIRCRAFT DESIGN

4.1 Locked Choice Optimization

To fully evaluate the algorithms capability to produce a viable aircraft, two test cases were run solely based on cost. In each case the discrete phenotype variables in the design string were locked using equality constraints. This cuts the design space into a more limited partition. All design runs are set to run ten times until 10000 generations with no improvement or when a quarter of a million generations have been generated. The objective function for these cases is given as

$$O = -\frac{CPA}{1000} - G \tag{4.1}$$

Where CPA is cost per aircraft and G is penalty from the constraint violations.

The missions listed in the following case studies are primarily payload hauls at minimum cost however in the unrestricted cases immediately following these cost only analyses the mission types vary based on performance requirements like takeoff and landing distance, maximum velocity, range, endurance. The resulting best design of ten trial runs is then displayed as well as a plot of the change in the leader’s objective function value as the generations progress.

4.1.1 Design of Traditional Turboprop V-Tail Group 4

Mission: Low Cost Group 4 UAV Operating at 5000 m

Weight Payload 500 kg

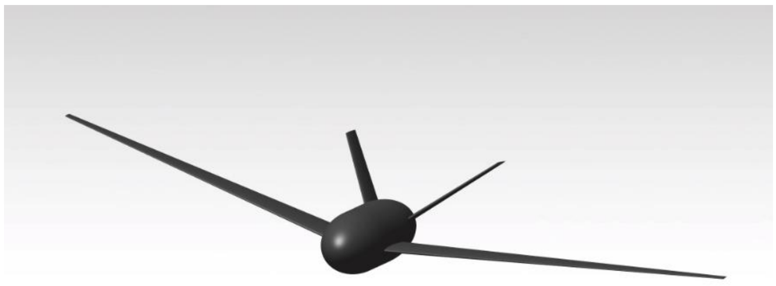
Constraints: Traditional Planform Only Gene 1 set to 1

V-Tail Only Gene 11 set to 3

Turboprop Only Gene 18 set to 2

Single Engine Gene 19 set to 1

The resulting design is shown in Figure 4.1 with its accompanying design string.



Group 4 Cost Optimization Design String	1
	6.6925(m)
	3.763(m)
	0.122(rad)
	0.476(rad)
	0.2029
	0.2070 (m)
	1.2012 (m)
	1.4998 (m)
	1.0642 (m)
	3
	0
	0
	0
	2.4 (m)
	.500
	0.3887
	2
	1
	458 (kW)

Figure 4.1 Low Cost Group 4 UAV and Design String

The aircraft defined above has the following performance evaluation

Table 4.1 Low Cost Group 4 UAV Evaluation

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax	Cost per Aircraft (millions USD)
3000	1821	1875	27.2	3.0	2	31	9.5

Attempting to conserve cost the algorithm severely cut the size of the engine mounted within the fuselage. In so doing the maximum velocity greatly suffered when compared to the turboprop Reaper. The wings themselves are incredibly thin and tapered. They are also severely swept. This appears to have resulted from attempting to provide a requisite static margin of 15% exploiting sweep to push the aerodynamic center behind the center of gravity rather than increase the length of various fuselage sections. Additionally, four of the ten runs exploited a weak constraint on tail sizing taking the constraint penalty of 2000 and gaining the value of the tail instead of putting the requisite tail or any tail at all. This is demonstrated in the divergence displayed in Figure 4.2.

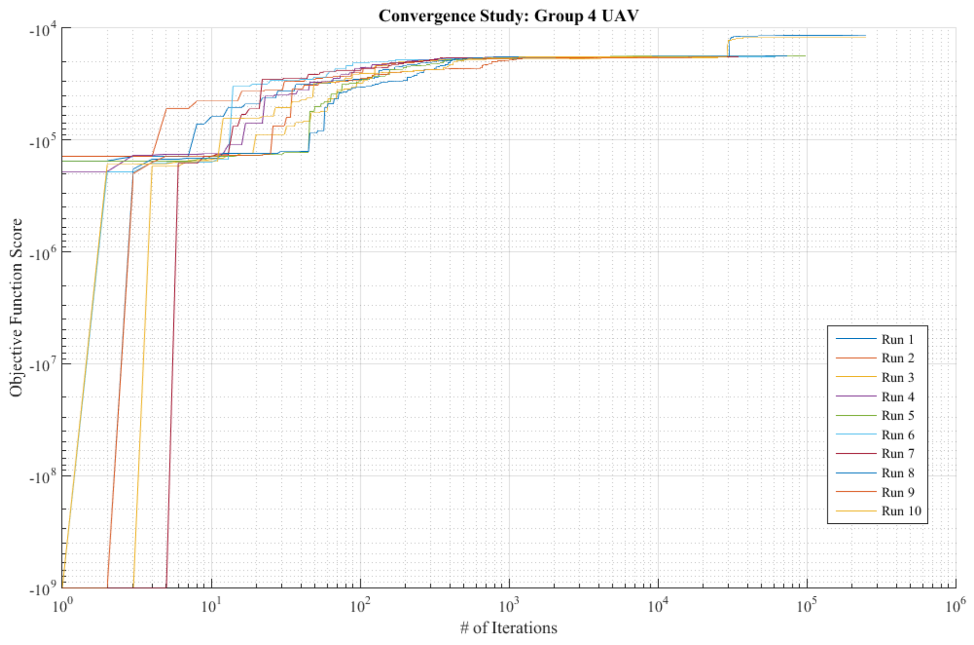


Figure 4.2 Group 4 Low Cost UAV Convergence

4.1.2 Traditional Turbofan V-Tail Group 5

Mission: Stable Low Cost Group 5 UAV Operating at 10000 m

Weight Payload 1700 kg

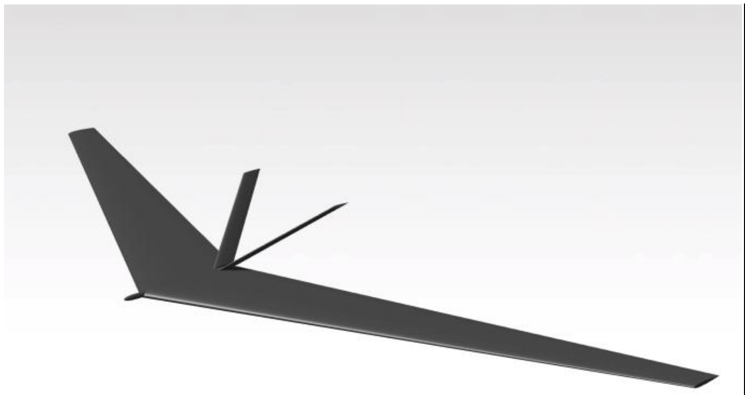
Constraints: Traditional Planform Only Gene 1 set to 1

V-Tail Only Gene 11 set to 3

Turbofan Only Gene 18 set to 3

Single Engine Gene 19 set to 1

The Resulting aircrafts and its design string can be seen in Figure 4.3.



Group 5 Cost Optimization Design String
1
19.99 (m)
4.491 (m)
0.1243(rad)
0.1924(rad)
0.3256
0.2040 (m)
1.0036 (m)
3.7769 (m)
0.5001 (m)
3
0.2102 (m)
0.0317 (m)
0.9160
4.6606 (m)
1.0244 (m)
0.8498
2
1
40 (kN)

Figure 4.3 Low Cost Group 5 UAV and Design String

The performance details of the aircraft shown in Figure 4.3 are given in Table 4.2

Table 4.2 Low Cost Group 4 UAV Evaluation

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax	Cost per Aircraft (1e6 USD)
18010	12006	8313	10.3	4.1353	8.274	72.26	67.8

This aircraft is almost exactly two thirds fuel by weight, likely lending to its high range and endurance. Uniquely the wing dominates the fuselage to the point where the basic shape resembles more of a flying wing than a traditional planform aircraft. Again, the stability is achieved from wing sweep and not sizing of the fuselage. The engine size is comparable to the Global Hawk though it travels at a significantly slower velocity, in part due to its lower operational altitude. The fuselage diameter took the brunt of the cost optimization, as the fuselage surface area is among the driving cost factors. Despite this exploitation, the aircraft is definitely feasible, however, with such a limited internal volume it is unlikely a continued design would be permitted to have such a small free space available. The convergence can be seen in Figure 4.4

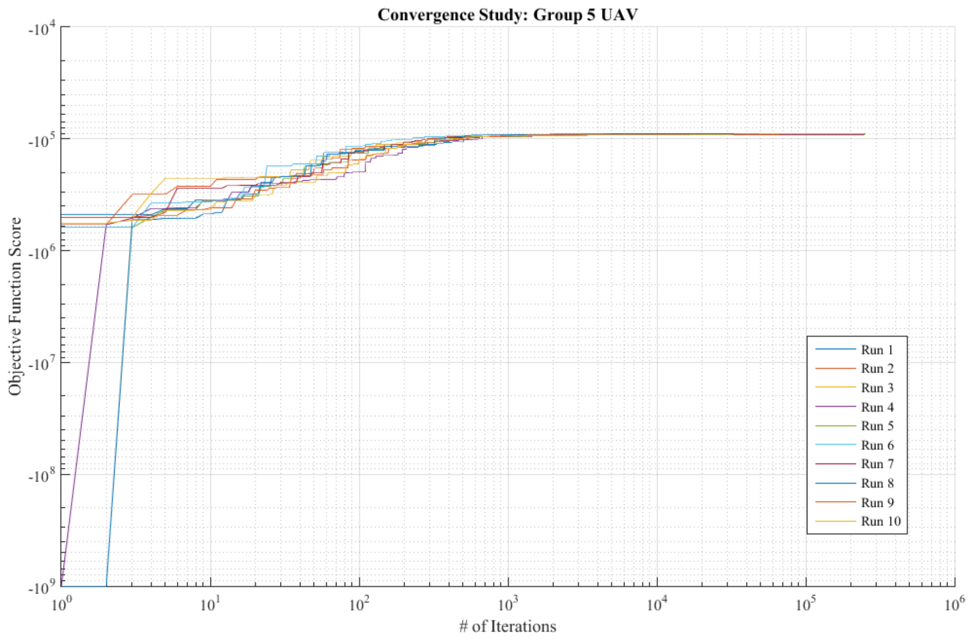


Figure 4.4 Low Cost Group 5 UAV Convergence

4.2 Free Space Optimization

The following designs were allowed complete use of the design space. The objective function is also changed to incorporate various performance metrics depending on the mission parameters. In example a mission type similar to that of the Global Hawk would reward all designs with a range and endurance over the requested value. The objective functions are listed with the mission type in each case study.

4.2.1 Group 4 UAV

The objective function for this optimization routine is defined as:

$$O = \frac{Range-3000}{1000} + Endurance - 5 + \frac{Vmax-120}{100} - \frac{CPA}{1000} - G \quad (4.2)$$

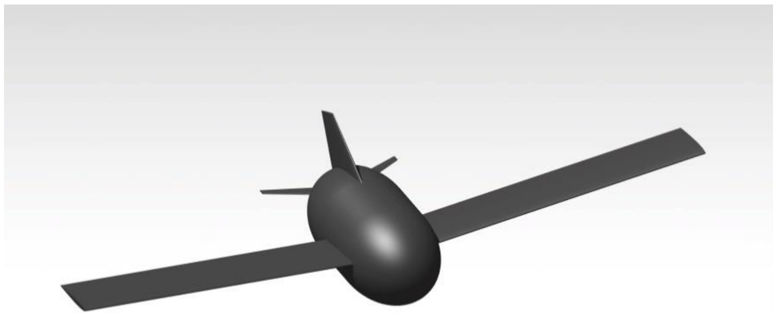
Mission: Low Cost High Speed Group 4 UAV

Range: 3000 km

Endurance: 5 hours enforced as a hard constraint

Vmax: > 120 m/s

The aircraft and the resulting design string can be seen in Figure 4.5 its performance can be seen in Table 4.3.



Group 4 Cost Optimization Design String	1
	6.3589(m)
	0.6375(m)
	0.1252(rad)
	0.0192(rad)
	0.9949
	1.9099 (m)
	1.000 (m)
	1.3814 (m)
	1.3833 (m)
	1
	0.9925 (m)
	0.3349 (m)
	0.4003
	1.5063 (m)
	1.4838 (m)
	0.3502
	3
	1
	15 (kN)

Figure 4.5 Group 4 UAV and Design String

Table 4.3 Group 4 UAV Evaluation

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax	Cost per Aircraft (1e6 USD)
2664	1347	3117	3.0	3.27	2.4	52.2	22.9

The aircraft returned by the optimization routine is unable to complete its requested mission with an endurance of only three hours instead of the requested five triggering a constraint violation penalty. This brings into question the validity of the supposed range as even at its maximum velocity traveling for the entire time of its maximum endurance the aircraft can only realistically travel 564 kilometers. All ten cases repeat this discrepancy runs in which convergence is not achieved within the entirety of 250,000 iterations. This is the result of over constraint of the design space. The combination of weight limitations, altitude requirements, and performance requirements, combined with the geometry constraints created an untenable design space. This failure to find a solution can be seen in Figure 4.6.

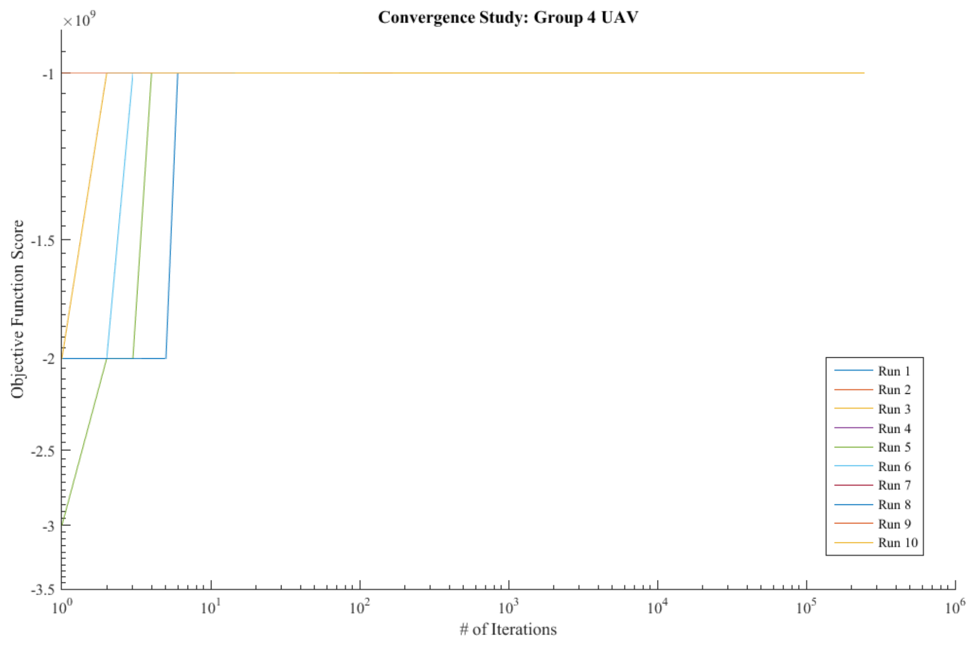


Figure 4.6 Group 4 UAV Convergence

4.2.2 Stable Stealth Group 5 UAV

The objective function for this case is defined as:

$$O = \frac{Range - 12000}{1000} + Endurance - 8 + \frac{Vmax - 150}{100} - RCS * 10 - \frac{CPA}{1000} - G \qquad (4.3)$$

Mission: Group 5 Low Cost Stable Stealth Long Range

Weight Payload: 1250 kg

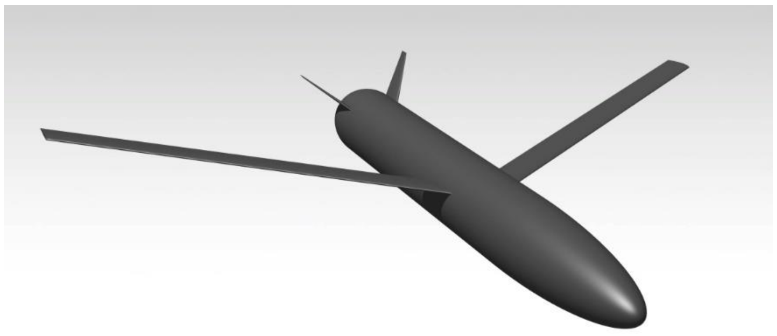
Range: 12000 km

Endurance: 8 hours

Vmax: > 150 m/s

Static Margin> 15% enforced as a constraint

With the addition of a stability constraint the possibility of a flying wing is greatly diminished. The resulting aircraft can be seen in Figure 4.7.



Group 4 Cost Optimization Design String	1
	9.7439(m)
	0.9825(m)
	0.1257(rad)
	0.5236(rad)
	0.9834
	1.709 (m)
	8.203 (m)
	4.444 (m)
	1.0734 (m)
	3
	0
	0
	0
	1.742 (m)
	.0.8519 (m)
	0.3524
	3
	1
	27 (kN)

Figure 4.7 Stealth Group 5 UAV and Design String

Table 4.4 Stealth Group 5 UAV Performance

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax (m/s)	RCS dB m^2	Cost per Aircraft (1e6 USD)
11122	5068	19854	16.08	29.47	22.7	240	45.6	140.5

The resulting aircraft looks less like a typical UAV and more akin to a long-range cruise missile. Comparing the traits, a long sleek fuselage, narrow swept wings, high maximum speed, high range and endurance, small control surfaces and more than half of its weight is fuel, the result is more pronounced in its favoritism towards this missile planform, but the basic traits can be observed in the cost only assessment of the Group 4 UAVs as well. The convergent behavior of this case is very different than the locked choice cases Figure 4.8. Instead the objective function varies wildly as different discrete variables are tried in combination often resulting in harsh constraint penalties and then quickly rebounding. Eventually the algorithm stabilizes and follows the more traditional convergence previously seen in the cost only cases.

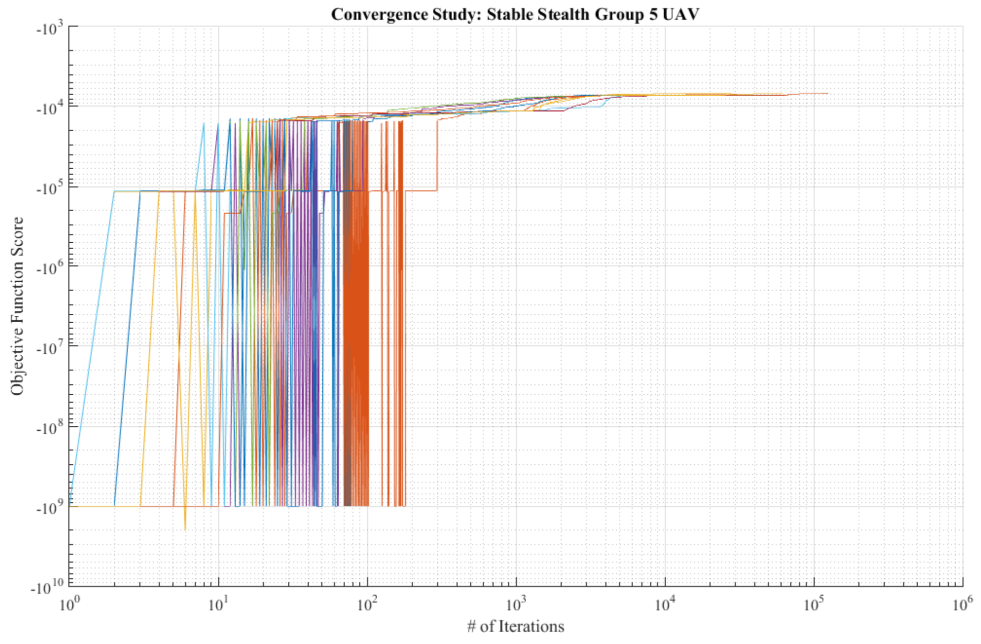


Figure 4.8 Stealth Group 5 UAV Convergence

4.3 Additional Design Concepts

4.3.1 Flying Wing Twin Engine Stealth Bomber

When running the stealth case above the solver will attempt to reconcile static margin. Naturally flying wings have difficulty obtaining high to moderate static margins by simple geometry. The same type of objective function was run again this time with no constraint placed on stability and added emphasis on stealth.

Mission: Stealth Group 5 UAV

$$O = \frac{Range - 7000}{1000} + Endurance - 4 + \frac{Vmax - 120}{100} - \cdots$$

$$\ldots RCS * 1000 - \frac{CPA}{1000} - G \qquad (4.4)$$

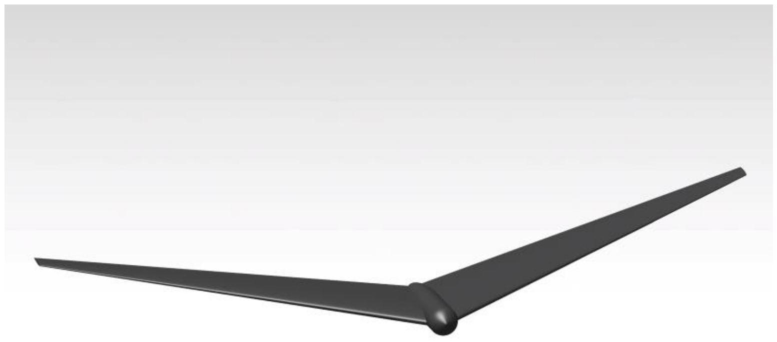
Max velocity> 120 m/s

Range > 7000 km

Endurance > 4 hours

Weight Payload = 2200 kg

The resulting aircraft and its design string are displayed in Figure 4.9.



Group 4 Cost Optimization Design String	3
	14.95(m)
	4.929(m)
	0.1257(rad)
	0.5236(rad)
	0.2
	0.404 (m)
	2.615 (m)
	0.68 (m)
	1.637 (m)
	1
	0
	0
	0
	0
	0
	0
	3
	2
	22 (kN)

Figure 4.9 Stealth Flying Wing UAV and Design String

The performance of the resulting aircraft is given in Table 4.5.

Table 4.5 Stealth Flying Wing UAV Evaluation

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax (m/s)	RCS dB m^2	Cost per Aircraft (1e6 USD)
17997	11998	7332	5.5	7.8	8.94	129	40.08	83.2

This aircraft possesses a stealthier airframe than the stable stealth Group 5 UAV, however its range and endurance are significantly lower likely do to the inclusion of two engines. However, with the inclusion of a heavier weighted stealth consideration the twin engines are a viable solution. The geometry of the engines is included in the projected area that reflects radar signals, therefore two smaller engines despite the increased cost was chosen over a single large engine to reduce the radar response. This case converged very quickly most likely because of the heavy stealth weighting limiting the design space to flying wings almost immediately. This convergence behavior is shown in Figure 4.10.

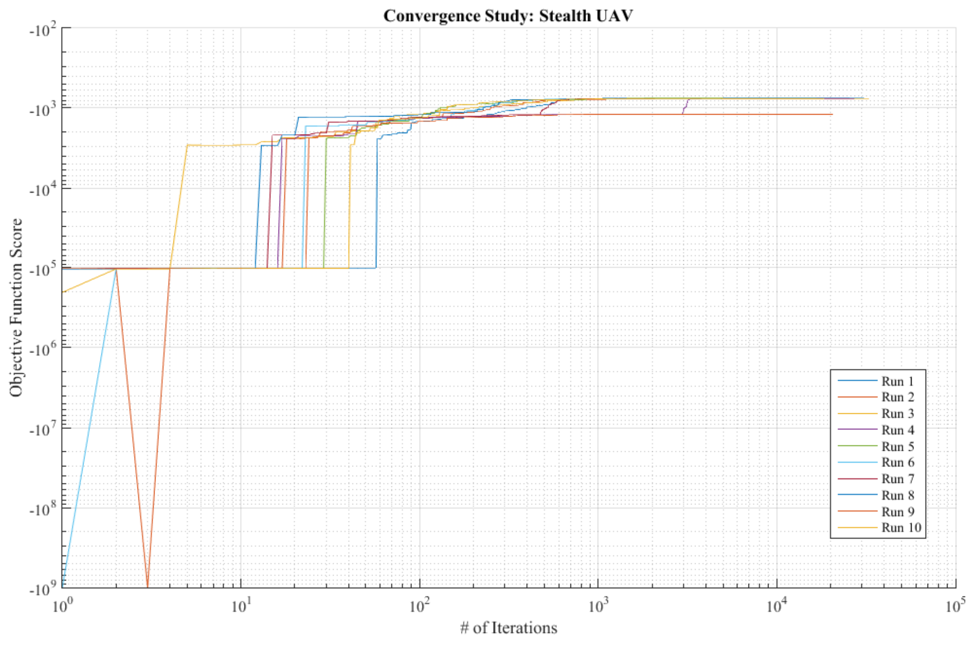


Figure 4.10 Stealth Flying Wing UAV Convergence

4.3.2 Short Field Mid-Range Ground Attacker

Many UAVs perform strike missions from forward operating bases. These bases have significantly shorter runways adding an additional constraint to the design space.

Mission: Short Range Forward Operating Attacker

$$O = \frac{Range - 3000}{1000} + \frac{Vmax - 140}{100} - G \qquad (4.5)$$

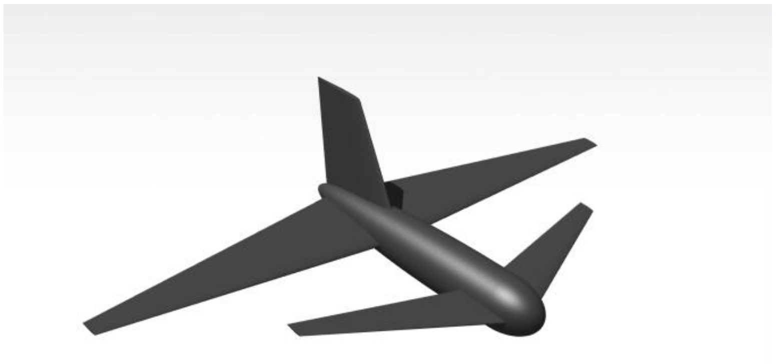
Max velocity> 140 m/s

Range > 3000 km

Weight Payload = 1750 kg

Runway Length <2000m

The resulting aircraft and its design string are displayed in Figure 4.11.



Group 4 Cost Optimization Design String	
	2
	8.34(m)
	2.23(m)
	0.0179(rad)
	0.000 (rad)
	0.200
	1.840 (m)
	1.463 (m)
	1.278 (m)
	4.993 (m)
	1
	4.156(m)
	1.382(m)
	0.56
	2.94 (m)
	2.46 (m)
	.970
	3
	2
	50 (kN)

Figure 4.11 Fast Attack Canard UAV and Design String

The aircrafts performance evaluation can be found in Table 4.6

Table 4.6 Fast Attack Canard UAV Evaluation

Weight (kg)	Weight Fuel (kg)	Range (km)	Endurance (hours)	Rate of Climb (m/s)	L/D max	Vmax (m/s)	Cost per Aircraft (1e6 USD)
11092	4146	3510	1.0	63.83	3.66	184	83.2

Due to the added constraint of the shortened runway the aircraft returned by the optimization cycle has two 50 kN engines. These provide a rate of climb of near 64 m/s. While this increase in thrust accommodates the short runway the velocity condition coupled with the twin engines drains fuel very quickly. The range requirement was met and the aircrafts profile does suggest that it fulfills the mission of a forward operating quick strike UAV. The convergence of this objective function is not similar to any one case. The objective function again bounces as in the stealth bomber case; however, the bouncing continues throughout the optimization routine. The objective function was shifted down by a value of 5000 to accommodate displaying a logarithmic plot of the designs improvement. The convergence plot is displayed in Figure 4.12.

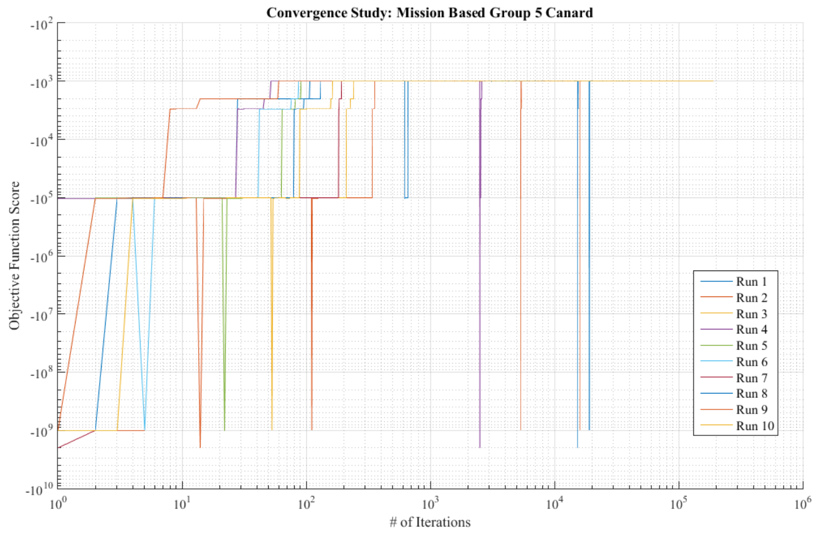


Figure 4.12 Fast Attack Canard UAV Convergence

CHAPTER 5

CONCLUSION

5.1 Conclusion

The algorithm presented in this research has demonstrated a capacity to synthesize an UAV from a set of mission requirements using cost and performance metrics in its optimization routine. The reliability of the Modified Breeder Pool Routine has also been demonstrated. While the algorithm is capable of providing a feasible aircraft, it cannot replicate the knowledge of an experienced designer, and as such should be considered for use only under the direct supervision of an experienced aircraft designer. While the proposed algorithm can in fact augment the design period and provide a reasonable starting point for the continuance of the design process it suffers from three major flaws.

The DAPCA IV cost model can be applied as a rough estimate for unmanned aerial vehicle program cost and per vehicle cost, but the implementation of a more rigorous model explicitly encompassing the extreme size range of UAVs would benefit the accuracy of the cost forecast.

The results of the algorithm layout a major shortcoming in aerodynamic force prediction, particularly with lift. The flat plate assumption used as the basis for the aerodynamic buildup is too simplified even for this preliminary case. A method implementing the use of airfoil sections even in the two-dimensional case would provide a new layer of accuracy in this early conceptual design phase.

Lastly the majority of the aircraft designed by the algorithm have insufficient volume to fully perform the mission in a reasonable capacity. To counter this, additional constraints on the total volume and the useable volume of the aircraft need to be implemented. This would ensure that the returned aircraft can hold its assigned payload and still have room for fuel and other systems and look more similar to typical aircraft.

5.2 Future Work

The genetic optimization routine implemented within will serve as a building block for more sophisticated versions of this routine. The end goal being to provide aircraft designers a way to immediately see design tradeoffs regarding choices in both discrete and continuous geometries. With the refined results from the improved version of the optimization routine, further optimization using derivative based methods can be implemented with the basic geometry output from the genetic algorithm. If implemented successfully the design process could be shortened substantially and the mission performance of the final aircraft greatly increased.

REFERENCES

[1] Anderson, John D. *Aircraft Performance and Design*. Boston: WCB/McGraw-Hill, 1999.

[2] Arora, Jasbir S. *Introduction to Optimum Design*. 3rd ed. Boston, MA: Academic, 2011.

[3] Callero, Monti. *Assessment of Nonlethal Unmanned Aerial Vehicles for Integration with Combat Aviation Missions*. Briefing. Santa Monica: RAND Corporation, 1995.

[4] Dauwalter, Charles R., and E. Russ Althof, eds. *AIAA Aerospace Design Engineers Guide*. 6th ed. Reston, VA: American Institute of Aeronautics and Astronautics, 2012.

[5] Elbeltagi, Emad, Tarek Hegazy, and Donald Grierson. "Comparison among Five Evolutionary-based Optimization Algorithms." *Advanced Engineering Informatics* 19.1 (2005): 43-53.

[6] Lewontin, Richard. "The Genotype/Phenotype Distinction." *Stanford University*. Stanford University, 23 Jan. 2004.

[7] Kenway, Gaetan K. W., and Joaquim R. R. A. Martins. "Multipoint High-Fidelity Aerostructural Optimization of a Transport Aircraft Configuration." *Journal of Aircraft* 51.1 (2014): 144-60.

[8] Kontogiannis, Spyridon G., and John A. Ekaterinaris. "Design, Performance Evaluation and Optimization of a UAV." *Aerospace Science and Technology* 29.1 (2013): 339-50.

[9] Martins, Joaquim R. R. A., and Andrew B. Lambe. "Multidisciplinary Design Optimization: A Survey of Architectures." *AIAA Journal* 51.9 (2013): 2049-075.

[10] McConnico, John Beck, and Patrick W. Moore. "Reciprocating Engines." *WADE: World Alliance for Decentralized Energy*. WADE, Jan. 2006.

[11] Niţă, Mihaela, and Dieter Scholz. *Estimating the Oswald Factor from Basic Aircraft Geometrical Parameters*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2012.

[12] Raymer, Daniel P. *Aircraft Design: A Conceptual Approach*. 5th ed. Reston, VA: American Institute of Aeronautics and Astronautics, 2012.

[13] Raymer, Daniel P. *Enhancing Aircraft Conceptual Design Using Multidisciplinary Optimization*. Thesis. Royal Institute of Technology, 2002. Stockholm: Tekniska Högsk., 2002.

[14] Samoylovitch, O., and D. Strelets. "Determination of the Oswald Efficiency Factor at the Aeroplane Design Preliminary Stage." *Aircraft Design* 3 (2000): 167-74.

[15] Sobester, Andras, and Andy J. Keane. "Multidisciplinary Design Optimization of UAV Airframes (AIAA)." *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Structures*. Newport, Rhode Island, May 2006.

[16] Tianyuan, Hu, and Yu Xiongqing. "Aerodynamic/Stealthy/Structural Multidisciplinary Design Optimization of Unmanned Combat Air Vehicle." *Chinese Journal of Aeronautics* 22.4 (2009): 380-86.

[17] UAS Task Force. *Department of Defense UAS Integration Plan*. Rep. no. 1-7ABA52E. Washington D.C.: United States Department of Defense, 2011.

[18] USAF. "MQ-1B Predator Fact Sheet." *MQ-1B Predator Fact Sheet Display*. United States Air Force, 2015. Web. 28 Oct. 2016.

[19] USAF. "MQ-9 Reaper Fact Sheet." *MQ-9 Reaper Fact Sheet Display*. United States Air Force, 2015. Web. 28 Oct. 2016.

[20] USAF. "RQ-4 Fact Sheet." *RQ-4 Global Hawk Fact Sheet Display*. United States Air Force, 2014. Web. 28 Oct. 2016.

[21] Weatherington, Dyke. *Unmanned Aircraft Systems*. Issue brief no. 10-S-1660. Washington D.C.: United States Department of Defense, 2010.

APPENDIX A

Performance Equations from [1][4][11]

$$Range\ Prop = \frac{1}{C} * \frac{L}{D} * \ln\left(\frac{W_o}{W_f}\right) \quad \text{(A.1)}$$

$$Range\ Jet = \frac{2}{C} * \frac{\sqrt{\frac{2}{\rho S}} C_L^{\frac{1}{2}}}{C_D} * \left(W_o^{\frac{1}{2}} - W_1^{\frac{1}{2}}\right) \quad \text{(A.2)}$$

$$Endurance\ Prop = \frac{1}{C} * \frac{\sqrt{2\rho S} C_L^{\frac{3}{2}}}{C_D} * \left(W_1^{-\frac{1}{2}} - W_0^{-\frac{1}{2}}\right) \quad \text{(A.3)}$$

$$Endurance\ Jet = \frac{1}{C} * \frac{L}{D} * \ln\left(\frac{W_o}{W_f}\right) \quad \text{(A.4)}$$

$$\left(\frac{L}{D}\right) \max = \frac{1}{2} * \sqrt{\frac{\pi e AR}{C_{D0}}} \quad \text{(A.5)}$$

$$Vstall = \sqrt{(2 * W * 1)/(\rho * S * C_L)} \quad \text{(A.6)}$$

$$Rate\ of\ Climb = \frac{R}{C} = \frac{T*V - D*V}{W} = (excess\ power)/W \quad \text{(A.7)}$$

$$Velocity\ of\ max\ turn = [\frac{2*\left(\frac{W}{S}\right)}{\rho}]^{1/2} * \left(\frac{K}{C_{D0}}\right)^{\frac{1}{4}} \quad \text{(A.8)}$$

$$\max turn\ load = nmax = (\frac{\frac{T}{W}}{\sqrt{K C_{D0}}} - 1)^{1/2} \quad \text{(A.9)}$$

$$\max turn\ rate = \omega = \frac{1}{2} * \rho * V^2 * \sqrt{\frac{\rho}{\frac{W}{S}} * \left[\frac{\frac{T}{W}}{2K} - \left(\frac{C_{D0}}{K}\right)^{\frac{1}{2}}\right]} \tag{A.10}$$

$$Radius\ of\ minimum\ turn = Rmin = \left(4K * \frac{\frac{W}{S}}{(g*\rho*\left(\frac{T}{W}\right)* \sqrt{1-\frac{4KC_{D0}}{\left(\frac{T}{W}\right)^2}}}\right) \tag{A.11}$$

$$takeoff\ ground\ roll = s_g \approx \frac{1.21*\left(\frac{W}{S}\right)}{\left(g\rho*C_{Lmax}\left(\frac{T}{W}\right)\right)} \tag{A.12}$$

$$landing\ distance = 1.1 * Vstall + \frac{1.1^2*\left(\frac{W}{S}\right)}{g\rho C_{Lmax}\left[\frac{D}{W}+\mu_r\left(1-\frac{L}{W}\right)\right]} \tag{A.13}$$

APPENDIX B

MATLAB Code

Main Driver Script

```
%geneticmainscript

close all
clc
n=1;
while n<=10
tic
z=1;
m=0;
itermax=250000;
lscore=zeros(itermax+1,1);
lavera=zeros(itermax+1,1);
lindex=zeros(itermax+1,1);
convergence=false;
START=zeros(1,20,500);
format long
A=getstart(START);
AIRCRAFT=A;
AIRCRAFT=getcorrect(AIRCRAFT);

[O,Av,G]=evaluate(AIRCRAFT);
[maxO,maxI]=max(O);
[C,D]=sort(O,'descend');
leader=AIRCRAFT(:,:,maxI);
lscore(z)=maxO;
lindex(z)=maxI;
lavera(z)=Av;
while convergence==false
    if mod(m,1000)==0
        AIRCRAFT(1,:,1)=leader;
        AIRCRAFT(1,:,2:end)=getstart(START(:,:,1:end-1));
        AIRCRAFT=getcorrect(AIRCRAFT);
    end
    if z>=2
        [O,Av,G]=evaluate(AIRCRAFT);
        [maxO,maxI]=max(O);
        leader=AIRCRAFT(:,:,maxI);
           lscore(z)=maxO;
        lindex(z)=maxI;


       lavera(z)=Av;
            if lscore(z)>lscore(z-1)
                 m=0;
```

```
                  leader=AIRCRAFT(:,:,maxI);
                 [C,D]=sort(O,'descend');
            elseif lscore(z)<=lscore(z-1)
                    m=m+1;
                    leader=AIRCRAFT(:,:,maxI);
                    [C,D]=sort(O,'descend');

            end
    end
    if z>itermax || (m>=10000 && G(1)==0)
        convergence=true;
    end
    [AIRCRAFT]=BreederPool(leader,AIRCRAFT,START,D);

AIRCRAFT=getcorrect(AIRCRAFT);
   z=z+1;

end
toc

[O,Range,Endurance,RCmax,RCmaxalt,edih,W_S,TorP_W,LDmax,CLmax,Vmax,Vmax
Range,VmaxEndurance,Vstall,nMaxRate,MinTurnRadius,StaticMargin,RCS,Cost
,CPA,Swet,Dblade,Deng,Heng,Leng,G,W,WFuel,nmaxturn,AR]=evaluatefinal(AI
RCRAFT);
 R(n)=Range(1);
 E(n)=Endurance(1);
 Vm(n)=Vmax(1);
 Vr(n)=VmaxRange(1);
 Ve(n)=VmaxEndurance(1);
 Co(n)=Cost(1);
 CPAn(n)=CPA(1);
 B(:,:,n)=leader;
 Score(:,n)=lscore;
 Vio(n)=G(1)
 n=n+1
```

Function getstart

```
function [START]=getstart(AIRCRAFT)
for i=1:length(AIRCRAFT)
    %Design String
    AIRCRAFT(1,1,i)=randi(3);
%AC Type 1
    %Wing Variables
    AIRCRAFT(1,2,i)=2+rand*18;
%Semispan 1 (m) 2
    AIRCRAFT(1,3,i)=0.5+7.5*rand;
%Root Chord (m) 3
    AIRCRAFT(1,4,i)=-pi./25+rand.*pi/12.5;
%Dihedral   (rad) 4
    AIRCRAFT(1,5,i)=-pi./4+rand.*2*pi/4;
%Sweep c/4  (rad) 5
    AIRCRAFT(1,6,i)=.25+rand*.75;
%Taper Span 6
    %Fuselage Variables
    AIRCRAFT(1,7,i)=.2+3.8*rand;
%Diameter Fuselage (m) 7
    AIRCRAFT(1,8,i)=.5+rand.*9;
%Length Main Fuselage (m) 8
    AIRCRAFT(1,9,i)=1.3*AIRCRAFT(1,7,i)*rand;
%Length Nose Section  (m) 9
    AIRCRAFT(1,10,i)=3.8*AIRCRAFT(1,7,i)*rand;
%Length Tail Section  (m) 10
    %Tail Variables
    AIRCRAFT(1,11,i)=randi(3);
%Tail Type Conventional-1 T-Tail-2 V-Tail 45 deg-3   11
    %Horizontal Tail
    AIRCRAFT(1,12,i)=1/4.*AIRCRAFT(1,2,i);
%Horizontal Tail Span (m) 12
    AIRCRAFT(1,13,i)=1+3*rand;
%Horizontal Root Chord (m) 13
    AIRCRAFT(1,14,i)=.35+.55*rand;
%Taper Ratio Horizontal Tail 14
    %Vertical Tail
    AIRCRAFT(1,15,i)=1/4.*AIRCRAFT(1,2,i);
%Vertical Tail Span (m) 15
    AIRCRAFT(1,16,i)=1+2*rand;
%Vertical Root Chord (m) 16
    AIRCRAFT(1,17,i)=.35+.55*rand;
%Taper Ratio Vertical Tail 17

    %Engine Variables



    AIRCRAFT(1,18,i)=randi(3);     %1 piston,2 turboprop,3 turbofan 18
```

```
  AIRCRAFT(1,19,i)=randi(2);
%number of engines limit 2 19
    AIRCRAFT(1,20,i)=15+rand.*3700;
%power or thrust per engine 20


end
START=AIRCRAFT;
end
```

function getcorrect

```
function [AIRCRAFT]=getcorrect(A)
%Function Returns feasible values for aircraft based on design string
%choices type engine type number of engines and type of tail
A(:,1,logical(A(1,1,:)<1))=1;
A(:,1,logical(A(1,1,:)>3))=3;
A(:,11,logical(A(1,11,:)<1))=1;
A(:,11,logical(A(1,11,:)>3))=3;
A(:,18,logical(A(1,18,:)<1))=1;
A(:,18,logical(A(1,18,:)>3))=3;
A(:,19,logical(A(1,19,:)<1))=1;
A(:,19,logical(A(1,19,:)>2))=2;
%limit choice constraints
A(:,1,logical(A(1,1,:)~=2))=2;
% A(:,11,logical(A(1,11,:)~=3))=3;
% A(:,18,logical(A(1,18,:)~=3))=3;
% A(:,19,logical(A(1,19,:)~=2))=1;

TTail=logical(A(:,11,:)==2);
A(:,13,TTail)=A(:,16,TTail(:)).*A(:,17,TTail(:));

ACc=logical(A(:,1,:)==2);
A(:,11,ACc(:))=1;

ACw=logical(A(:,1,:)==3);
A(:,11,ACw(:))=1;
A(:,12,ACw(:))=0;
A(:,13,ACw(:))=0;
A(:,14,ACw(:))=0;
A(:,15,ACw(:))=0;
A(:,16,ACw(:))=0;
A(:,17,ACw(:))=0;
A(A(:,5,ACw(:))>50)=50*pi./180;


A(A(:,5,ACw(:))>50)=15.*pi./180;
APo=logical(A(1,18,:)==1 & A(1,20,:)>200);
A(:,20,APo(:))=200;
```

```
APu=logical(A(1,18,:)==1 & A(1,20,:)<75);
A(:,20,APu(:))=75;


ATo=logical(A(1,18,:)==2 & A(1,20,:)>3600);
A(:,20,ATo(:))=3600;

ATu=logical(A(1,18,:)==2 & A(1,20,:)<370);
A(:,20,ATu(:))=370;

ATFo=logical(A(1,18,:)==3 & A(1,20,:)>300);
A(:,20,ATFo(:))=300;

ATFu=logical(A(1,18,:)==3 & A(1,20,:)<10);
A(:,20,ATFu(:))=10;
A(logical(A(1,:,:)<0))=.1;


AIRCRAFT=A;


end

function BreederPool


function [ AIRCRAFT ] = BreederPool( leader,A,START,D)
AIRCRAFT=START;
RStart=zeros(1,20,100);
mut=2;
AIRCRAFT(1,:,1)=leader(1,:,1);
AIRCRAFT(:,:,2:1:100)=A(:,:,D(2:1:100));
for n=101:1:400
    s=randi(19)+1;
    m=randi(100);
    q=randi(100);

AIRCRAFT(:,1:s-1,n)=AIRCRAFT(:,1:s-1,m);
AIRCRAFT(:,s:end,n)=AIRCRAFT(:,s:end,q);
 for d=1:1:mut
        x=randi(2);
        z=randi(20);
        r=randi(100);
        if x==1

            if AIRCRAFT(1,z,n)==0



                AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1/r;
            elseif z==1
                AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==11
                 AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==18
                 AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==19
                  AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            else
                AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n).*1/r+AIRCRAFT(1,z,n);
            end
        elseif x==2

             if AIRCRAFT(1,z,n)==0
                AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1/r;
            elseif z==1
                AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==11
                 AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==18
                 AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
            elseif z==19
                  AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n)+1;
             else
                 AIRCRAFT(1,z,n)=AIRCRAFT(1,z,n).*(-
1)./r+AIRCRAFT(1,z,n);
            end
            end
        end
 end

AIRCRAFT(:,:,401:1:500)=getstart(RStart);

end
```

Function getSs

```
function [S,Sexposed,Sfuse,Shorz,Svert,Sfusen,Sfusem,Sfuset]=getSs(A)
indvtail=find(A(1,11,:)==3);
S=A(1,2,:).*A(1,3,:).*(1+A(1,6,:));
%S Wing b/2*cr*(1+taper)
Sexposed=S-(A(1,7,:)./2).*A(1,3,:).*(1+A(1,6,:));
%Sexposed Wing
Sfusem=2.*pi.*(A(1,7,:)./2).*A(1,8,:);
%Sfuse main




Sfusen=((pi.*A(1,7,:)./2)./(6.*A(1,9,:)).*(((pi.*A(1,7,:)./2).^2+4.*A(1
,9,:).^2).^(3/2)-(A(1,7,:)./2).^3));     %Sfuse nose
Sfuset=((pi.*A(1,7,:)./2)./(6.*A(1,10,:)).*(((pi.*A(1,7,:)./2).^2+4.*A(
1,10,:).^2).^(3/2)-(A(1,7,:)./2).^3));   %Sfuse tail
```

```
Swingcut=-2.*(.12.*(A(1,3,:)).*.8);
%SfuseWing intersect
Sfuse=Sfusem+Sfusen+Sfuset+Swingcut;
Shorz=A(1,12,:).*A(1,13,:).*(1+A(1,14,:));
Svert=1/2.*A(1,15,:).*A(1,16,:).*(1+A(1,17,:));
Svert(indvtail(:,1))=Svert(indvtail(:,1)).*2;
end
```

Function getSwet

```
function
[Swetwing,Swethorz,Swetvert,Swet]=getSwet(Sexp,Shorz,Svert,Sfuse)
tcwing=.12;
tctail=.12;
Swetwing=Sexp.*(1.977+0.52.*tcwing);
Swethorz=Shorz.*(1.977+0.52.*tctail);
Swetvert=Svert.*(1.977+0.52.*tctail);
Swet=Swetwing+Sfuse+Swethorz+Swetvert;
end
```

Function getAspectRatio

```
function[AR,AReff,ARvert,ARhorz]=getAspectRatio(A,S,Svert,Shorz)

AR=((2.*A(1,2,:)).^2)./S;     %aspect ratio
beff=2.*(A(1,2,:)./cos(A(1,4,:)));
Seff=beff./2.*A(1,3,:).*(1+A(1,6,:));
AReff=beff.^2./Seff;
clear beff
clear Seff
ARvert=((2*A(1,15,:).^2))./Svert;
ARhorz=((2*A(1,12,:).^2))./Shorz;
end
```

Function getEnginesize

```
function [ Weng,Leng,Deng,Heng,SFC,Dblade] = getEnginesize( A )




%Engine Size and Characteristics calculated from Aircraft Design String
A

BPR=2;
%Bypass Ratio TurboFan
```

```
ind1=find(A(:,18,:)==1);
%piston indices 2770 rpm 45-225 kW
ind2=find(A(:,18,:)==2);
%turboprop indices 370-3728 kW
ind3=find(A(:,18,:)==3);
%turbofan indices 15000-300000 N max Thrust


%Prop inline
Weng(:,:,ind1(:,1))=2.98.*(A(:,20,ind1(:,1))).^0.780;
%kg
Leng(:,:,ind1(:,1))=0.17.*(A(:,20,ind1(:,1))).^0.424;
%m
Deng(:,:,ind1(:,1))=0.5;
%m
Heng(:,:,ind1(:,1))=0.5;
%m
SFC(:,:,ind1(:,1))= 0.068.*A(:,19,ind1(:,1));
%Cbhp mg/W-s
Dblade(:,:,ind1(:,1))=0.52.*(A(:,20,ind1(:,1))).^0.25;
%m

%Turboprop
Weng(:,:,ind2(:,1))=0.96.*(A(:,20,ind2(:,1))).^0.803;
%kg
Leng(:,:,ind2(:,1))=0.12.*(A(:,20,ind2(:,1))).^0.373;
%m
Deng(:,:,ind2(:,1))=0.25.*(A(:,20,ind2(:,1))).^0.120;
%m
Heng(:,:,ind2(:,1))=0;
%m
SFC(:,:,ind2(:,1))=0.085.*A(:,19,ind2(:,1));
%Cbhp mg/W-s
Dblade(:,:,ind2(:,1))=0.52.*(A(:,20,ind2(:,1))).^0.25;
%m

%Turbofan
Weng(:,:,ind3(:,1))=14.7.*(A(:,20,ind3(:,1))).^1.1.*exp(-0.045.*BPR);
%kg
Leng(:,:,ind3(:,1))=0.49.*(A(:,20,ind3(:,1))).^0.4.*0.9.^0.2;
%m
Deng(:,:,ind3(:,1))=0.15.*(A(:,20,ind3(:,1))).^0.5.*exp(0.04.*BPR);
%m
Heng(:,:,ind3(:,1))=0;
%equivalent to diameter
```

```
SFC(:,:,ind3(:,1))=22.7.*A(:,19,ind3(:,1));
%mg/Ns
Dblade(:,:,ind3(:,1))=0;
%no blades present
clear ind1 ind2 ind3 BPR
end
```

function getVolumes

```
function [Vfuse,VFuel,TotalVolume,VWing,Vcut] =getVolumes(A)

tc=.12;
h=A(1,2,:);                  %a b c d represent sides of bases of
pyrimidal frustrum h represents the height
a=0.65.*A(1,3,:);            %root
b=A(1,3,:).*tc ;             %root thickness
c=0.75.*A(1,3,:).*A(1,6,:); %tip
d=A(1,3,:).*A(1,6,:).*tc;   %tip thickness
VWing=2/3.*h.*((a.*b).^2+(a.*b).*(c.*d)+(c.*d).^2);
eta=(A(1,7,:)./2)./h;       %normalized spanstation location of edge of
fuselage
cfuse=(A(1,3,:).*(1-eta.*(1-A(1,6,:))));
cthick=cfuse.*tc;
cfuse=0.65.*cfuse;
Vcut=2/3.*A(1,7,:)./2.*((a.*b).^2+(a.*b).*(cfuse.*cthick)+(cfuse.*cthic
k).^2);
Vfuse=(pi.*(A(1,7,:)./2).^2.*A(1,8,:))+(1/2.*pi.*(A(1,7,:)./2).^2.*A(1,
9,:))+(1/2.*pi.*(A(1,7,:)./2).^2.*A(1,10,:))-Vcut;
cutbig=find(Vfuse(:,:,:)<0);
Vfuse(:,:,cutbig(:,1))=0;
VFuel=1/2.5.*VWing+1/4.*Vfuse;
TotalVolume=Vfuse+VWing;
clear h a b c d eta cfuse cthick cutbig
end
```

function getDragBuildup

```
function[CDo,MACs,quarterwing,quarterhorz,quartervert]=getDragBuildup(A
,S,Swetwing,Sfuse,Shorz,Svert,Dblade,Deng,rho)
ind3=logical(A(1,1,:)==3);

%Find Indices of Different Tails
TT1(:,1)=find(A(:,11,:)==1);
TT2(:,1)=find(A(:,11,:)==2);
TT3(:,1)=find(A(:,11,:)==3);
PT12(:,1)=find(A(:,18,:)~=3);
```

```
PT3(:,1)=find(A(:,18,:)==3);

%Approximate method for Drag Utilizing Wing Only
Cfeapprox=.0030;
% CDoapprox=Cfeapprox.*(Sexposed./S);

%Drag Build Up Method

%Get Form Factors FF
FFwing(1,1,:)=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cos(A(1,5,:)))
);                                                          %Wing Form
Factor
FFhorz(1,1,:)=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cosd(35)));
%Horizontal Stabalizer Form Factor
FFvert(1,1,:)=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cos(45)));
%Vertical Stabalizer Form Factor
f=A(1,8,:)./A(1,7,:);
%Fuselage Fineness Ratio
FFfuse(1,1,:)=(1+60./f.^3+f./400);
%Fuselage Form Factor

%Get Interference Factor Component Q
Qwing(1,1,1:1:length(A))=1.0;
%Wing Interference Factor
Qfuse(1,1,1:1:length(A))=1.0;
%Fuselage Interference Factor
Qhorz(1,1,TT1(:,1))=1.05;
%Traditional Tail Interference Factors
Qvert(1,1,TT1(:,1))=1.05;
Qhorz(1,1,TT2(:,1))=1.08;
%T-Tail Interference Factor
Qvert(1,1,TT2(:,1))=1.08;
Qhorz(1,1,TT3(:,1))=1.03;
%V-Tail Interference Factor
Qvert(1,1,TT3(:,1))=1.03;

%Get Mean Aerodynamic Chords
[MACs,quarterwing,quarterhorz,quartervert]=getMACs(A);

%Get Reynbolds Numbers
REwing(1,1,:)=(rho.*100.*MACs(1,:,:))./(1.73*10^-5);
%Wing Reynolds Number
REhorz(1,1,:)=(rho.*100.*MACs(2,:,:))./(1.73*10^-5);
%Horizontal Stabalizer Reynolds
REvert(1,1,:)=(rho.*100.*MACs(3,:,:))./(1.73*10^-5);
%Vertical Stabalizer Reynolds
REfuse(1,1,:)=(rho.*100.*(A(1,8,:)+A(1,9,:)+A(1,10,:)))./(1.73*10^-5);
%Fuselage Reynolds Number
%Get Coefficient of Friction Cf Component
```

```
Cfwing(1,1,:)=0.455/((log(REwing(1,1,:)).^2.58).*(1+0.144.*.4.^2).^0.65
);                                                          %Wing Friction
Coeficient
Cfhorz(1,1,:)=0.455/((log(REhorz(1,1,:)).^2.58).*(1+0.144.*.4.^2).^0.65
);                                                          %Horizontal
Stabalizer Friction Coeficient
Cfvert(1,1,:)=0.455/((log(REvert(1,1,:)).^2.58).*(1+0.144.*.4.^2).^0.65
);                                                          %Vertical
Stabalizer Friction Coeficient
Cffuse(1,1,:)=0.455/((log(REfuse(1,1,:)).^2.58).*(1+0.144.*.4.^2).^0.65
);                                                          %Fuselage
Friction Coeficient

Cfhorz(1,1,ind3(:))=0;
Cfvert(1,1,ind3(:))=0;

%Get CDo

if isempty(PT12)

DragDisk(1,1,PT3(:))=Cfeapprox.*(0.25.*(pi.*(Deng(:,:,PT3(:,1))./2).^2)
).*A(1,19,PT3(:,1));
elseif isempty(PT3)

DragDisk(1,1,PT12(:))=Cfeapprox.*(0.33.*(pi.*(Dblade(:,:,PT12(:,1))./2)
.^2)).*A(1,19,PT12(:,1));
else

DragDisk(1,1,PT12(:))=Cfeapprox.*(0.33.*(pi.*(Dblade(:,:,PT12(:))./2).^
2)).*A(1,19,PT12(:));                                 %Propulsion Drag
Propeller Disk

DragDisk(1,1,PT3(:))=Cfeapprox.*(0.25.*(pi.*(Deng(:,:,PT3(:))./2).^2)).
*A(1,19,PT3(:));                                      %Propulsion Drag
Jet Inlet
end


CDo(1,1,:)=((Cfwing(1,1,:)).*FFwing(1,1,:).*Qwing(1,1,:).*Swetwing(1,1,
:))+...

(Cfhorz(1,1,:).*FFhorz(1,1,:).*Qhorz(1,1,:).*Shorz(1,1,:))+(Cfvert(1,1,
:).*FFvert(1,1,:).*Qvert(1,1,:).*Svert(1,1,:))+...   %CDo Calculation

(Cffuse(1,1,:).*FFfuse(1,1,:).*Qfuse(1,1,:).*Sfuse(1,1,:))./S(1,1,:)+Dr
agDisk(1,1,:);


CDo(1,1,:)=CDo(1,1,:).*1.05;
%Add Leakage and Protuberance Drag Estimation
```

```
clear FFwing FFhorz FFvert FFfuse Qwing Qfuse Qhorz Qvert REhorz REvert
REfuse CFwing Cfhorz Cfvert Cffuse DragDisk
end
```

Function getMACs

```
function [MACs,quarterwing,quarterhorz,quartervert]=getMACs(Aircraft)
indfw=logical(Aircraft(1,1,:)==3);
A=Aircraft(1,3,:);
B=Aircraft(1,3,:).*Aircraft(1,6,:);
MACwing=A-(2.*(A-B).*(0.5.*A+B))./(3.*(A+B));
C=Aircraft(1,13,:);
D=Aircraft(1,13,:).*Aircraft(1,14,:);
MAChorz=C-(2.*(C-D).*(0.5.*C+D))./(3.*(C+D));
E=Aircraft(1,16,:);
F=Aircraft(1,16,:).*Aircraft(1,17,:);
MACvert=E-(2.*(E-F).*(0.5.*E+F))./(3.*(E+F));
MACvert(1,1,indfw(:))=0;
MAChorz(1,1,indfw(:))=0;
quarterwing=.25*MACwing(:,:,:);
quarterhorz=.25.*MAChorz(:,:,:);
quartervert=.25.*MACvert(:,:,:);
MACs=[MACwing(:,:,:);MAChorz(:,:,:);MACvert(:,:,:)];
end
```

function getCG

```
function
[cgnosef,cgmainf,cgtailf,cgwing,qcwing,cgvert,cghorz,armhorz,armvert]=g
etCG(A,MACs)
%cg location 0 is nose tip moving positive towards tail section
ind1=find(A(1,1,:)==1);
ind2=find(A(1,1,:)==2);
ind3=find(A(1,1,:)==3);
TT1=find(A(1,11,:)==1);
TT2=find(A(1,11,:)==2);
TT3=find(A(1,11,:)==3);

cgnosef=2./3.*A(1,9,:);
cgmainf=A(1,9,:)+.5.*A(1,8,:);
cgtailf=A(1,8,:)+A(1,9,:)+1./3.*A(1,10,:);
cgwing(1,1,ind1(:))=A(1,9,ind1(:))+.3.*A(1,8,ind1(:))+MACs(1,:,ind1(:))
./2.*sin(A(1,5,ind1(:)))+0.4.*MACs(1,:,ind1(:));
cgwing(1,1,ind2(:))=A(1,9,ind2(:))+.7.*A(1,8,ind2(:))+MACs(1,:,ind2(:))
./2.*sin(A(1,5,ind2(:)))+0.4.*MACs(1,:,ind2(:));
```

```
cgwing(1,1,ind3(:))=.25.*A(1,9,ind3(:))+MACs(1,:,ind3(:))./2.*sin(A(1,5
,ind3(:)))+0.4.*MACs(1,:,ind3(:));

qcwing(1,1,ind1(:))=A(1,9,ind1(:))+.3.*A(1,8,ind1(:))+MACs(1,:,ind1(:))
./2.*sin(A(1,5,ind1(:)))+0.25.*MACs(1,:,ind1(:));
qcwing(1,1,ind2(:))=A(1,9,ind2(:))+.8.*A(1,8,ind2(:))+MACs(1,:,ind2(:))
./2.*sin(A(1,5,ind2(:)))+0.25.*MACs(1,:,ind2(:));
qcwing(1,1,ind3(:))=.30.*A(1,9,ind3(:))+MACs(1,:,ind3(:))./2.*sin(A(1,5
,ind3(:)))+0.25.*MACs(1,:,ind3(:));

cgvert(1,1,ind1(:))=sind(30).*MACs(3,:,ind1(:))./2+A(1,9,ind1(:))+A(1,8
,ind1(:))+.40.*A(1,10,ind1(:))+0.4.*MACs(3,:,ind1(:));
cgvert(1,1,ind2(:))=sind(30).*MACs(3,:,ind2(:))./2+A(1,9,ind2(:))+A(1,8
,ind2(:))+.40.*A(1,10,ind2(:))+0.4.*MACs(3,:,ind2(:));
cgvert(1,1,ind3(:))=sind(30).*MACs(3,:,ind3(:))./2+A(1,9,ind3(:))+A(1,8
,ind3(:))+.40.*A(1,10,ind3(:))+0.4.*MACs(3,:,ind3(:));

qcvert(1,1,ind1(:))=sind(30).*MACs(3,:,ind1(:))./2+A(1,9,ind1(:))+A(1,8
,ind1(:))+.25.*A(1,10,ind1(:))+0.25.*MACs(3,:,ind1(:));
qcvert(1,1,ind2(:))=sind(30).*MACs(3,:,ind2(:))./2+A(1,9,ind2(:))+A(1,8
,ind2(:))+.25.*A(1,10,ind2(:))+0.25.*MACs(3,:,ind2(:));
qcvert(1,1,ind3(:))=sind(30).*MACs(3,:,ind3(:))./2+A(1,9,ind3(:))+A(1,8
,ind3(:))+.25.*A(1,10,ind3(:))+0.25.*MACs(3,:,ind3(:));

cghorz(1,1,TT1(:))=sind(20).*MACs(2,:,TT1(:))./2+A(1,9,TT1(:))+A(1,8,TT
1(:))+.40.*A(1,10,TT1(:))+0.4.*MACs(3,:,TT1(:));
cghorz(1,1,TT2(:))=A(1,15,TT2(:)).*sind(30)+sind(20).*MACs(2,:,TT2(:)).
/2+A(1,9,TT2(:))+A(1,8,TT2(:))+.40.*A(1,10,TT2(:))+0.4.*MACs(3,:,TT2(:)
);
cghorz(1,1,TT3(:))=sind(20).*MACs(2,:,TT3(:))./2+A(1,9,TT3(:))+A(1,8,TT
3(:))+.40.*A(1,10,TT3(:))+0.4.*MACs(3,:,TT3(:));

s=find((A(1,11,:)==2) & (A(1,1,:)==2));
cghorz(1,1,s(:))=.60.*A(1,9,s(:))+sind(20).*MACs(2,:,s(:))./2+0.4.*MACs
(3,:,s(:));

qchorz(1,1,TT1(:))=sind(20).*MACs(2,:,TT1(:))./2+A(1,9,TT1(:))+A(1,8,TT
1(:))+.25.*A(1,10,TT1(:))+0.25.*MACs(3,:,TT1(:));
qchorz(1,1,TT2(:))=A(1,15,TT2(:)).*sind(30)+sind(20).*MACs(2,:,TT2(:)).
/2+A(1,9,TT2(:))+A(1,8,TT2(:))+.25.*A(1,10,TT2(:))+0.25.*MACs(3,:,TT2(:
));
qchorz(1,1,TT3(:))=sind(20).*MACs(2,:,TT3(:))./2+A(1,9,TT3(:))+A(1,8,TT
3(:))+.25.*A(1,10,TT3(:))+0.25.*MACs(3,:,TT3(:));

qchorz(1,1,s(:))=.60.*A(1,9,s(:))+sind(20).*MACs(2,:,s(:))./2+0.25.*MAC
s(3,:,s(:));

armhorz(1,1,:)=abs((qcwing(:)-qchorz(:)));
armvert(1,1,:)=abs((qcwing(:)-qcvert(:)));
```

function getTailVolumeCoef

```
function
[cvt,cht]=getTailVolumeCoef(armhorz,armvert,A,Svertwet,Shorzwet,Swet)
ind3=(logical(A(1,1,:)==3));
TT3=(logical(A(1,11,:)==3));
b=A(1,2,:);
c=A(1,3,:);

cvt=(armvert.*Svertwet)./(b.*Swet);
cht=(armhorz.*Shorzwet)./(c.*Swet);

cvt(TT3(:))=((armvert(TT3(:)).*Svertwet(TT3(:))).*sind(45))./(b(TT3(:))
.*Swet(TT3(:)));
cht(TT3(:))=((armvert(TT3(:)).*Svertwet(TT3(:))).*sind(45))./(c(TT3(:))
.*Swet(TT3(:)));

cvt(ind3(:))=0;
cht(ind3(:))=0;


end
```

function getCLalpha

```
function [CLalpha]=getCLalpha(A,AR,S,Sxp)
F=1.07*(1+A(1,7,:)./(2.*A(1,2,:))).^2;   %Fuselage Lift Contribution
and Interference Factor F
Betasqr=1-.4^2;      %Compresibility Correction
eta=.95;             %Mach Correlation Airfoil Efficiency use 0.95 or 1
too ignore all together
CLalpha=((2*pi.*AR)./(2+sqrt(4+(AR.^2+Betasqr)./(eta.^2)).*(1+(tan(A(1,
5,:))).^2))).*(Sxp./S).*F;

end
```

Function getoswald

```
function [e,e_dih]=getoswald(A,AR,AReff,CLalpha)

Beta=sqrt(1-.4.^2);
[Se]=getSuction(A,AR);
[kf]=getKf(A);
[e]=(1/(((2.*AR)./(CLalpha)).*(1-Se)+Se)).*kf;
e_dih=AReff./AR.*e;
end
```

function getSuction

```
function [Se]=getSuction(A,AR)
Se=0.974-0.0976.*exp(-0.456.*((AR.*A(1,6,:))./(cos(A(1,5,:)))));
end
```

function getKf

```
function [kf]=getKf(A)
dbar=[0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1];
factor=[1,.98,.95,.92,.84,.75,.64,.5,.35,.18,0];
kf=interp1(dbar,factor,(A(1,7,:)./(2.*A(1,2,:))),'linear');

end
```

function getWeight

```
function
[W,Wempty,WFuel,Wwing,Whtail,Wvtail,Wengines,Wpayload,Wauxiliary]=getWe
ight(A,Sexp,Sfuse,Shorz,Svert,VFuel,Weng)
%Weight Approximation Function for Design String

Wpayload=1360;
Wauxiliary=150;
Wwing=Sexp.*20;
Whtail=Shorz.*12;
Wvtail=Svert.*12;
Wfuselage=Sfuse.*15;
ind1=find(A(1,18,:)==1);
```

```
ind2=find(A(1,18,:)==2);
ind3=find(A(1,18,:)==3);
WFuel(:,:,ind1(:,1))=720.*VFuel(:,:,ind1(:,1));
WFuel(:,:,ind2(:,1))=800.*VFuel(:,:,ind2(:,1));
WFuel(:,:,ind3(:,1))=800.*VFuel(:,:,ind3(:,1));
Wengines(1,1,:)=1.3.*Weng(1,1,:).*A(1,19,:);
Wempty(1,1,:)=(Wwing+Whtail+Wvtail+Wfuselage)+Wengines;
W=Wempty(1,1,:)+WFuel(1,1,:)+Wpayload+Wauxiliary;
W=W.*0.033+W;
end
```

Function getW_SandTP_W

```
function [ W_S,W_Sexp,TorP_W,TorP_Walt] = getW_SandTP_W( A,W,S,Sexp,rho
)

id1=find((A(1,18,:)==1));
id2=find((A(1,18,:)==2));
id3=find((A(1,18,:)==3));
W_S=W.*(9.81)./S;
W_Sexp=W.*(9.81)./Sexp;
TorP_W=((A(1,19,:).*A(1,20,:)).*1000)./(W.*9.81);

    TorP_Walt(1,1,id1(:))=TorP_W(id1(:)).*(1.132.*(rho./1.225)-0.132);
    TorP_Walt(1,1,id2(:))=TorP_W(id2(:)).*((rho./1.225).^0.7);
    TorP_Walt(1,1,id3(:))=TorP_W(id3(:)).*(rho./1.225).^0.6;
clear id1 id2 id3
end
```

function getSteadyLevel

```
function
[Vmax,Vstall,Vtip,Vstallalt]=SteadyLevel(A,S,W,CDo,K,CLmax,rho,TorP_W,W
_S,Dblade,a)
ind1=find(A(1,18,:)~=3);
ind3=find(A(1,18,:)==3);
Vstall(:,:,:)=sqrt((2./1.225).*(W./S).*(1./CLmax));
Vstallalt(:,:,:)=sqrt((2./rho).*(W./S).*(1./CLmax));

%Prop Maximum Velocity
Vtip=pi.*2770./60.*Dblade./2;
% a=1/2.*rho.*S(ind1(:)).*CDo(ind1(:));
% b=(2.*(W(ind1(:)).*9.81).^2.*K(ind1(:)))./(S(ind1(:)).*rho);
% c=-0.8.*((A(:,19,ind1(:)).*(A(1,20,ind1(:)))*1000));
% for i=1:1:length(ind1)
%     j=20;
%     while j<290
%     B(i,j)=a(i).*j.^3+b(i).*j.^(-1)+c(i);
%
%         if B(i,j)*B(i,j-1)<0
%         V(1,1,i)=1/2.*(j+(j-1));
%         end
%         j=j+1;
%     end
% end
if isempty(ind3)
[i,j]=meshgrid(1:1:length(ind1),20:0.5:290);
a2=1/2.*rho.*S(1,i(1,:)).*CDo(1,i(1,:));
b2=(2.*(W(1,i(1,:)).*9.81).^2.*K(1,i(1,:)))./((S(1,i(1,:))).*rho);
c2=-0.8.*((A(:,19,i(1,:)).*(A(1,20,i(1,:)))*1000));
B=a2(i).*j.^3+b2(i).*j.^-1+c2(i);

for k=1:1:length(ind1);

    if isnan(B(:,k))
        V(1,1,k)=0;
    elseif isempty(find(B(:,k)>0,1,'first'))
           V(1,1,k)=0;
    else
    [q(k)]=find(B(:,k)>0,1,'first');
%    q
        if q(k)==1;

              V(1,1,k)=20.5;

        else
             V(1,1,k)=(q(k)+q(k)-2).*1./2.*.5+20;

        end
    end
end

Vmax(:,:,ind1(:,1))=V(1,1,:);
Vmax(logical(sqrt(Vmax(ind1(:)).^2+Vtip(ind1(:)).^2)>=a))=.75.*a;

elseif isempty(ind1)
%Jet Maximum Velocity

Vmax(:,:,ind3(:,1))=sqrt(((TorP_W(:,:,ind3(:,1))).*W_S(:,:,ind3(:,1))+W
_S(:,:,ind3(:,1)).*sqrt((TorP_W(:,:,ind3(:,1))).^2-
4.*CDo(:,:,ind3(:,1)).*K(:,:,ind3(:,1)))))./(rho.*CDo(:,:,ind3(:,1))));
Vmax(logical(Vmax>0.8.*a))=0.8.*a;
Else
```

```
    [i,j]=meshgrid(1:1:length(ind1),20:0.5:290);
    a2=1/2.*rho.*S(1,i(1,:)).*CDo(1,i(1,:));

    b2=(2.*(W(1,i(1,:)).*9.81).^2.*K(1,i(1,:)))./((S(1,i(1,:))).*rho);

    c2=-0.8.*((A(:,19,i(1,:)).*(A(1,20,i(1,:)))*1000));

    B=a2(i).*j.^3+b2(i)./j+c2(i);


    for k=1:1:length(ind1);

    if isnan(B(:,k))
        V(1,1,k)=0;
    elseif isempty(find(B(:,k)>0,1,'first'))
           V(1,1,k)=0;
    else
    [q(k)]=find(B(:,k)>0,1,'first');
%    q
          if q(k)==1;

                V(1,1,k)=20.5;

          else
               V(1,1,k)=(q(k)+q(k)-2).*1./2.*.5+20;

          end
    end
end



Vmax(:,:,ind1(:,1))=V(1,1,:);
Vmax(logical(sqrt(Vmax(ind1).^2+Vtip(ind1).^2)>=a))=.75.*a;


Vmax(:,:,ind3(:,1))=sqrt(((TorP_W(:,:,ind3(:,1))).*W_S(:,:,ind3(:,1))+W
_S(:,:,ind3(:,1)).*sqrt((TorP_W(:,:,ind3(:,1))).^2-
4.*CDo(:,:,ind3(:,1)).*K(:,:,ind3(:,1))))./(rho.*CDo(:,:,ind3(:,1))));
Vmax(logical(Vmax>0.8.*a))=0.8.*a;
end
Vmax(isnan(Vmax))=0;
Vmax(logical(imag(Vmax)))=0;
%check imaginary speed
Vmax(Vmax<Vstallalt)=0;

clear ind1 ind3 x a2 b2 c2 B V q
end
```

Function getStaticMargin

```
function
[StaticMargin,CGx]=getStaticMargin(A,cgnosef,cgmainf,cgtailf,cgwing,cgv
ert,cghorz,Sfusen,Sfusem,Sfuset,WFuel,Whtail,Wengines,Wpayload,Wauxilia
ry,Wwing,Wvtail,MACs,qcwing,W)
ind1=logical(A(1,19,:)==1);
ind2=logical(A(1,19,:)==2);

CG=cgnosef.*Sfusen.*15+cgmainf.*Sfusem.*15+cgtailf.*Sfuset.*15+cgwing.*
Wwing+cghorz.*Whtail+cgvert.*Wvtail+cgwing.*.5.*WFuel+cgmainf.*.5.*WFue
l+Wpayload.*.9.*cgmainf+Wauxiliary.*cgnosef;
CG(ind1(:))=CG(ind1(:))+Wengines(ind1(:)).*.75.*cgtailf(ind1(:));
CG(ind2(:))=CG(ind2(:))+Wengines(ind2(:)).*1.25.*cgmainf(ind2(:));
CGx=CG./W;
StaticMargin=(qcwing-CGx)./MACs(1,:,:).*100;
End
```

Function getFormFactor

```
function [FFwing,FFhorz,FFvert,FFfuse]=getFormfactor(A)
FFwing=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cos(A(1,5,:))));
FFhorz=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cosd(35)));
FFvert=(1+(0.6/.3)*.12+100*.12^4).*(1.34*.4^.18.*(cos(45)));
f=A(1,8,:)./A(1,7,:)
FFfuse=(1+60./f.^3+f./400);
%

end
```

Function get LDmax

```
function [ L_Dmax, CLmax ] = getLDmax(K,CDo,CLalpha)
CDo(isnan(CDo))=60;
CDo(logical(imag(CDo)))=60;
K(logical(imag(K)))=1;

L_Dmax(:,:,:)=1/2.*(sqrt(1./(CDo(1,:,:).*K(1,:,:))));
L_Dmax(logical(isnan(L_Dmax)))=.1;
CLmax=(10.*pi/180).*CLalpha.*1.2;

end
```

function getPerformance

```
function
[VmaxRange,VmaxEndurance,Range,Endurance,RCmaxs,RCmaxalt]=getPerformanc
e(A,Wfuel,W,K,S,CDo,SFC,rho,Vmax,W_S,LDmax,TorP_W,TorP_Walt,Enduranceb)
indprop=find(A(1,18,:)~=3);
indjet=find(A(1,18,:)==3);
ind1=find(A(1,18,:)==1);
ind2=find(A(1,18,:)==2);
```

```
%Minimum Thrust
CL_CD12=3/4.*(1./(3.*K.*CDo.^3)).^(1/4);
V_CL12=((2./rho).*sqrt((3.*K)./(CDo)).*W_S).^(1/2);
TR12=W.*9.81./(CL_CD12);
Cmin12=SFC.*1e-6.*9.81;
%Minimum Power
CL_CD32=1/4.*(3./(K.*CDo.^(1./3))).^(3./4);
V_CL32=((2./rho).*(sqrt(K./(3.*CDo)).*W_S)).^(1/2);
TR32=(9.81.*W)./(CL_CD32);
Cmin32=SFC.*1e-6.*9.81;
%Max L/D
V_LDmax=((2./rho).*(sqrt(K./CDo)).*W_S).^(1/2);
TRmaxLD=(W.*9.81)./(LDmax);
CmaxLD=SFC.*1e-6.*9.81;



%Range
Range(1,1,indprop(:))=((0.8)./(CmaxLD(indprop(:)))).*(LDmax(indprop(:))
).*log(9.81.*W(indprop(:))./(9.81.*(W(indprop(:))-Wfuel(indprop(:))))));
Range(1,1,indjet(:))=(2./Cmin12(indjet(:))).*sqrt(2./(rho.*S(indjet(:))
)).*CL_CD12(indjet(:)).*((9.81.*W(indjet(:)).^(1/2))-
(9.81.*(W(indjet(:))-Wfuel(indjet(:)))).^(1/2));
Range=Range./1000;
Range(logical(isnan(Range)))=0;
Range(logical(imag(Range)))=0;

VmaxRange(:,:,indprop(:))=V_LDmax(indprop(:));
VmaxRange(:,:,indjet(:))=V_CL12(indjet(:));
%Endurance
Endurance(1,1,indprop(:))=(0.8./Cmin32(indprop(:))).*sqrt(2.*rho.*S(ind
prop(:))).*CL_CD32(indprop(:)).*((9.81.*(W(indprop(:))-
0.95.*Wfuel(indprop(:)))).^(-1/2)-(9.81.*W(indprop(:))).^(-1/2));
Endurance(1,1,indjet(:))=(1./CmaxLD(indjet(:))).*(LDmax(indjet(:))).*lo
g(9.81.*W(indjet(:))./(9.81.*(W(indjet(:))-0.95.*Wfuel(indjet(:)))));
Endurance=Endurance./3600;
Range(Range>30000)=30000;
% Endurance=1.14.*(Range.*1000./VmaxRange);
VmaxEndurance(:,:,indprop(:))=V_CL32(indprop(:));
VmaxEndurance(:,:,indjet(:))=V_LDmax(indjet(:));
Endurance(logical(isnan(Endurance)))=0;
Endurance(logical(imag(Endurance)))=0;
VmaxEndurance(isnan(VmaxEndurance))=0;
VmaxEndurance(logical(imag(VmaxEndurance)))=0;
VmaxRange(isnan(VmaxRange))=0;
VmaxRange(logical(imag(VmaxRange)))=0;

%Rate of Climb
%sealevel
Zs(1,1,:)=1+sqrt(1+3./(LDmax(:).^2.*TorP_W(:).^2));
```

```
Zalt(1,1,:)=1+sqrt(1+3./(LDmax(:).^2.*TorP_Walt(:).^2));
RCmaxs(1,1,indprop(:))=0.8.*TorP_W(indprop(:))-
(2./1.225.*sqrt(K(indprop(:))./(3.*CDo(indprop(:)))).*(W_S(indprop(:)))
).^(1/2).*(1.155./LDmax(indprop(:)));
RCmaxs(1,1,indjet(:))=((W_S(indjet(:)).*Zs(indjet(:)))./(3.*1.225.*CDo(
indjet(:)))).^(1/2).*(TorP_W(indjet(:))).^(3/2).*(1-Zs(indjet(:))./6-
(3./(2.*(TorP_W(indjet(:))).^2.*LDmax(indjet(:)).^2.*Zs(indjet(:)))));
%At requested altitude

    RCmaxalt(1,1,ind1(:))=0.8.*TorP_Walt(ind1(:))-
(2./rho.*sqrt(K(ind1(:))./(3.*CDo(ind1(:)))).*(W_S(ind1(:)))).^(1/2).*(
1.155./LDmax(ind1(:)));
    RCmaxalt(1,1,ind2(:))=0.8.*TorP_Walt(ind2(:))-
(2./rho.*sqrt(K(ind2(:))./(3.*CDo(ind2(:)))).*(W_S(ind2(:)))).^(1/2).*(
1.155./LDmax(ind2(:)));

RCmaxalt(1,1,indjet(:))=((W_S(indjet(:)).*Zalt(indjet(:)))./(3.*rho.*CD
o(indjet(:)))).^(1/2).*(TorP_Walt(indjet(:))).^(3/2).*(1-
Zalt(indjet(:))./6-
(3./(2.*(TorP_Walt(indjet(:))).^2.*LDmax(indjet(:)).^2.*Zalt(indjet(:))
)));

end

Function getRunway
function
[takeoff,landing]=getRunway(A,W_S,TorP_W,CL_max,Vstall,W,K,CDo,Sexp)
ind=(find(A(1,18,:)~=3));
if isempty(ind)
    takeoff=1.21.*(W_S)./(9.81.*CL_max.*TorP_W);
   V=1.1.*Vstall;
    V07=0.7.*V;

landing=V.*3+(1.1.^2.*W_S)./(9.81.*1.225.*CL_max.*(0+(1./2.*1.225.*V07.
^2.*(CDo+K.*(CL_max).^2))./(W.*9.81)+0.4.*(1-
(1./2.*1.225.*V07.^2.*CL_max.*Sexp)./(W.*9.81))));
else
V=1.1.*Vstall;

takeoff=1.21.*(W_S)./(9.81.*CL_max.*TorP_W);
takeoff(ind(:))=1.21.*(W_S(ind(:)))./(9.81.*CL_max(ind(:)).*TorP_W(ind(
:))./(V(ind(:))));
V07=0.7.*V;
landing=V.*3+(1.1.^2.*W_S)./(9.81.*1.225.*CL_max.*(0+(1./2.*1.225.*V07.
^2.*(CDo+K.*(CL_max).^2))./(W.*9.81)+0.4.*(1-
(1./2.*1.225.*V07.^2.*CL_max.*Sexp)./(W.*9.81))));
end

end
```

```
function getTurn
function
[MaxTurnV,MaxTurnRate,nMaxRate,MinTurnRadius,nmaxturn,TurnRadiusMaxV]=
getTurn(rho,TorP_Walt,Vmax,VmaxRange,VmaxEndurance,CLmax,K,CDo,W_S,A)
ind1=logical((A(1,18,:)~=3));
ind2=logical((A(1,18,:)==3));
%maximumturnrate
VmaxRate(1,1,:)=(2.*(W_S(:))./rho).^(1/2).*(K(:)./CDo(:)).^(1/4);

nMaxRate(1,1,ind1(:))=((TorP_Walt(1,1,ind1(:)).*1./VmaxRate(ind1(:)))./
(sqrt(K(ind1(:)).*CDo(ind1(:))))-1).^1/2;
nMaxRate(1,1,ind2(:))=((TorP_Walt(ind2(:))./(sqrt(K(ind2(:)).*CDo(ind2(
:)))))-1).^1/2);
MaxTurnRate(1,1,ind1(:))=1./2.*rho.*VmaxRate(ind1(:)).^2.*sqrt(rho./W_S
(ind1(:)).*((TorP_Walt(ind1(:)).*1./VmaxRate(ind1(:)))./(2.*K(ind1(:)))
-sqrt(CDo(ind1(:))./K(ind1(:)))));
MaxTurnRate(1,1,ind2(:))=1./2.*rho.*VmaxRate(ind2(:)).^2.*sqrt(rho./W_S
(ind2(:)).*((TorP_Walt(ind2(:)))./(2.*K(ind2(:)))-
sqrt(CDo(ind2(:))./K(ind2(:)))));
%minimumTurnRadius
MinTurnRadius=VmaxRate.^2./(9.81.*sqrt(nMaxRate.^2-1));
%m


%maximum load factor
nmaxturn=1./2.*rho.*Vmax.^2.*CLmax./W_S;
nmaxturn(nmaxturn>3.5)=3.5;
TurnRadiusMaxV=Vmax.^2./(9.81.*sqrt(nmaxturn.^2-1));


MaxTurnV=0;
nmaxneg=-2;
MaxTurnRate(isnan(MaxTurnRate))=.1;
MaxTurnRate(logical(imag(MaxTurnRate)))=.1;




end




function getRCS

function [RCS,Bottom]=getRCS(A,Sexp,Shorz,Svert,Dblade,Deng,Leng)
indv=logical(A(1,11,:)==3);
Asidenose=2./3.*2.*A(1,7,:).*A(1,9,:);
Amainside=2.*A(1,7,:).*A(1,8,:);
Asidetail=2./3.*2.*A(1,7,:).*A(1,10,:);
```

```
Range=1000;
RPower=10000;
Ptarget=RPower./Range.^2;
%side profile
sidewing=1/2.*Sexp.*sin(A(1,4,:))+.12.*A(1,3,:).*A(1,6,:);
sidehorz=.12.*1./2.*(A(1,13,:).*A(1,14,:)+A(1,13,:)).*sind(30);
sidevert=Svert;
sidevert(indv(:))=Svert(indv(:))./2.*sind(45);
sideengine=Deng.*Leng.*1./3;
Awingcut=(.12.*A(1,3,:).*.5).*A(1,3,:);

%front profile
frontfuse=pi.*(A(1,7,:)).^2;
frontblade=(1./2.*Dblade).^2.*pi.*1./3.*A(1,19,:);
frontengine=((1./2.*Deng).*1./4).^2.*A(1,19,:);
frontwing=2.*A(1,2,:).*(.12.*A(1,3,:)+.12.*(A(1,3,:).*A(1,6,:))./2).*co
s(A(1,5,:));
frontvert=sind(30).*.12.*1./2.*(A(1,16,:)+A(1,16,:).*A(1,17,:)).*(A(1,1
5,:));
fronthorz=cosd(20).*2.*A(1,12,:).*.12.*1./2.*(A(1,13,:)+A(1,13,:).*A(1,
14,:));
frontvert(indv(:))=2.*sind(30).*.12.*1./2.*(A(1,16,indv(:))+A(1,16,indv
(:)).*A(1,17,indv(:))).*(A(1,15,indv(:)));

%bottom profile
bottomfuse=Asidenose+Amainside+Asidetail;
bottomwing=Sexp.*cos(A(1,4,:));
bottomtail=Shorz;
bottomtail(indv(:))=Shorz(indv(:)).*cosd(45);


Aside=Asidenose+Amainside+Asidetail+sidewing+sidehorz+sidevert+sideengi
ne;
Front=frontfuse+frontblade+frontengine+frontwing+frontvert+fronthorz+fr
ontvert-Awingcut;
Bottom=bottomfuse+bottomwing+bottomtail;

RCSside=4.*pi.*Range.^2.*(Aside.^2.*.05.*Ptarget)./RPower;
RCSFront=4.*pi.*Range.^2.*(Front.^2.*.05.*Ptarget)./RPower;
RCSBottom=4.*pi.*Range.^2.*(Bottom.^2.*.05.*Ptarget)./RPower;
RCS=[RCSside,RCSFront,RCSBottom];
RCS=10.*log(RCS./1);


end
```

Function getOBJValue

```
function[O,P]=getOBJValue(Vmax,Vstall,Vstallalt,StaticMargin,W_S,TorP_W
,takeoff,landing,W,WFuel,Range,Endurance,RCmax,nMaxRate,MaxTurnRate,Min
TurnRadius,Cost,CPA,G,RCS,AR,LDmax)
```

```
O=(Range-3000)./(3000)+(Endurance-5)./5+(Vmax-150)./150-G-1000*RCS;
P=sum(O)./length(O);
end

function getCPenalty
function
[G]=getCPenalty(A,AR,Vmax,Vstallalt,StaticMargin,W_S,TorP_W,cvt,cht,RCm
ax,RCmaxalt,takeoff,landing,W,Wfuel,Range,Endurance,Deng,a,VmaxEnduranc
e,VmaxRange,ARhorz,ARvert)
G=zeros(1,1,length(A));
minW_S=30;                              %kg/m^2
maxW_S=586;                             %kg/m^2
SMpos=15;                               %percent
SMneg=-5;                               %- percent
runway=2000;                            %m
FRW=.666667;                            %ratio Fuel to TOWeight
Rangemin=3000;                          %Requested Range km
Endurancemin=6;                         %Requested Endurance hr
ttclimbmin=700;                         %time to climb requested
CostReq=10000000;                       %Price per AIRCRAFT requested
SPmin=1;                                %Semispan min m
SPmax=20;                               %Semispan max m
RChmin=.25;                             %rootchord min m
RChmax=10;                              %rootchord max m
Dihmin=-pi/25;                          %dihedral min rad
Dihmax=pi/25;                           %dihedral max rad
Swmin=-pi/6;                            %Sweep min rad
Swmax=pi/6;                             %Sweep max rad
Tpmin=.2;                               %taper ratio min
Tpmax=1.0;                              %taper ratio max
Dfmin=0.2;                              %Fuselage min diameter m
Dfmax=4.0;                              %fuselage max diameter m
Lfmmin=1;                             %Length main min fuselage m
Lfmmax=10;                              %length main max fuselage m
Lfnmin=.5;                              %Length nose min fuselage m
Lfnmax=5;                               %Length nose max fuselage m
Lftmin=0.5;                             %Length tail min fuselage m
Lftmax=5;                               %Length tail max fuselage m
Horzsmin=0;                             %Horz Stab span min m
Horzsmax=5;                             %Horz Stab span max m
Horzcmin=0;                             %Horz Stab chord min m
Horzcmax=4;                             %Horz Stab chord max m
Htapermin=.35;                            %Horz Stab taper min
Htapermax=1;                            %Horz Stab taper max
Vertsmin=0;                             %Vert Stab span min m
Vertsmax=5;                             %Vert Stab span max m
Vertcmin=0;                             %Vert Stab chord min m
Vertcmax=4;                             %Vert Stab chord max m
Vtapermin=.35;                          %Vert Stab taper min
Vtapermax=1;                            %Vert Stab taper max
```

```
FuseL=A(1,8,:)+A(1,9,:)+A(1,10,:);
WingL=A(1,3,:)+(A(1,13,:)+A(1,16,:))./2;
%% Constrain Violation and Penalty Assesment
%Geometry Constrants

G(((AR<3)))=G((AR<3))+1e9;%(3-AR(logical(AR<3))).^2;
G((AR>20))=G((AR>20))+1e9;%((30-AR(logical(AR>30))).^2);
%Span Limit Main Wing
G((A(1,2,:)<SPmin))=G((A(1,2,:)<SPmin))+1e9;%+(SPmin-
A(1,2,((A(1,2,:)<1)))).^2;
G((A(1,2,:)>SPmax))=G((A(1,2,:)>SPmax))+1e9;%(SPmax-
A(1,2,((A(1,2,:)>20)))).^2;
%Chord Limit Main Wing
G((A(1,3,:)<RChmin))=G((A(1,3,:)<RChmin))+1e9;%(RChmin-
A(1,3,((A(1,3,:)<RChmin)))).^2;
G((A(1,3,:)>RChmax))=G((A(1,3,:)>RChmax))+1e9;%(RChmax-
A(1,3,((A(1,3,:)>RChmax)))).^2;
%Dihedral
G((A(1,4,:)<Dihmin))=G((A(1,4,:)<Dihmin))+1e9;%(Dihmin-
A(1,4,((A(1,4,:)<Dihmin)))).^2;
G((A(1,4,:)>Dihmax))=G((A(1,4,:)>Dihmax))+1e9;%(Dihmax-
A(1,4,((A(1,4,:)>Dihmax)))).^2;
%Sweep
G((A(1,5,:)<Swmin))=G((A(1,5,:)<Swmin))+1e9;%(Swmin-
A(1,5,((A(1,5,:)<Swmin)))).^2;
G((A(1,5,:)>Swmax))=G((A(1,5,:)>Swmax))+1e9;%(Swmax-
A(1,5,((A(1,5,:)>Swmax)))).^2;
%Taper Ratio
G((A(1,6,:)<Tpmin))=G((A(1,6,:)<Tpmin))+1e9;%(Tpmin-
A(1,6,((A(1,6,:)<Tpmin)))).^2;
G((A(1,6,:)>Tpmax))=G((A(1,6,:)>Tpmax))+1e9;%(Tpmax-
A(1,6,((A(1,6,:)>Tpmax)))).^2;
%Fuselage Diameter
G((A(1,7,:)<Dfmin))=G((A(1,7,:)<Dfmin))+1e9;%(Dfmin-
A(1,7,((A(1,7,:)<Dfmin)))).^2;
G((A(1,7,:)>Dfmax))=G((A(1,7,:)>Dfmax))+1e9;%(Dfmax-
A(1,7,((A(1,7,:)>Dfmax)))).^2;
%Fuselage length main
G((A(1,8,:)<Lfmmin))=G((A(1,8,:)<Lfmmin))+1e9;%(Lfmmin-
A(1,8,((A(1,8,:)<Lfmmin)))).^2;
G((A(1,8,:)>Lfmmax))=G((A(1,8,:)>Lfmmax))+1e9;%(Lfmmax-
A(1,8,((A(1,8,:)>Lfmmax)))).^2;
%Fuselage length nose
G((A(1,9,:)<Lfnmin))=G((A(1,9,:)<Lfnmin))+1e9;%(Lfnmin-
A(1,9,((A(1,9,:)<Lfnmin)))).^2;
G((A(1,9,:)>Lfnmax))=G((A(1,9,:)>Lfnmax))+1e9;%(Lfnmax-
A(1,9,((A(1,9,:)>Lfnmax)))).^2;
%Fuselage length tail
G((A(1,10,:)<Lftmin))=G((A(1,10,:)<Lftmin))+1e9;%(Lftmin-
A(1,10,((A(1,10,:)<Lftmin)))).^2;
```

```
G((A(1,10,:)>Lftmax))=G((A(1,10,:)>Lftmax))+1e9;%(Lftmax-
A(1,10,((A(1,10,:)>Lftmax)))).^2;
%Horizontal Stabalizer Span
G((A(1,12,:)<Horzsmin))=G((A(1,12,:)<Horzsmin))+1e9;%(Horzsmin-
A(1,12,((A(1,12,:)<Horzsmin)))).^2;
G((A(1,12,:)>Horzsmax))=G((A(1,12,:)>Horzsmax))+1e9;%(Horzsmax-
A(1,12,((A(1,12,:)>Horzsmax)))).^2;
%Horizontal Stabalizer Chord
G((A(1,13,:)<Horzcmin))=G((A(1,13,:)<Horzcmin))+1e9;%(Horzcmin-
A(1,13,((A(1,13,:)<Horzcmin)))).^2;
G((A(1,13,:)>Horzcmax))=G((A(1,13,:)>Horzcmax))+1e9;%(Horzcmax-
A(1,13,((A(1,13,:)>Horzcmax)))).^2;
%Horizontal Stabalizer Taper
G((A(1,14,:)<Htapermin)&(A(1,1,:)~=3))=G((A(1,14,:)<Htapermin)&(A(1,1,:
)~=3))+1e9;%(Htapermin-A(1,14,((A(1,14,:)<Htapermin)))).^2;
G((A(1,14,:)>Htapermax))=G((A(1,14,:)>Htapermax)&(A(1,1,:)~=3))+1e9;%(H
tapermax-A(1,14,((A(1,14,:)>Htapermax)))).^2;
%Vertical Stabalizer Span
G((A(1,15,:)<Vertsmin))=G((A(1,15,:)<Vertsmin))+1e9;%(Vertsmin-
A(1,15,((A(1,15,:)<Vertsmin)))).^2;
G((A(1,15,:)>Vertsmax))=G((A(1,15,:)>Vertsmax))+1e9;%(Vertsmax-
A(1,15,((A(1,15,:)>Vertsmax)))).^2;
%Vertical Stabalizer Chord
G((A(1,16,:)<Vertcmin))=G((A(1,16,:)<Vertcmin))+1e9;%(Vertcmin-
A(1,16,((A(1,16,:)<Vertcmin)))).^2;
G((A(1,16,:)>Vertcmax))=G((A(1,16,:)>Vertcmax))+1e9;%(Vertcmax-
A(1,16,((A(1,16,:)>Vertcmax)))).^2;
%Vertical Stabalizer Taper
G((A(1,17,:)<Vtapermin)&(A(1,1,:)~=3))=G((A(1,17,:)<Vtapermin)&(A(1,1,:
)~=3))+1e9;%(Vtapermin-A(1,17,((A(1,17,:)<Vtapermin)))).^2;
G((A(1,17,:)>Vtapermax))=G((A(1,17,:)>Vtapermax))+1e9;%(Vtapermax-
A(1,17,((A(1,17,:)>Vtapermax)))).^2;
%Tail Aspect Ratio
G(ARhorz<3 & (A(1,1,:)~=3))=G(ARhorz<3 & (A(1,1,:)~=3))+1e9;
G(ARvert<3 & (A(1,1,:)~=3))=G(ARvert<3 & (A(1,1,:)~=3))+1e9;
G(ARhorz>7 & (A(1,1,:)~=3))=G(ARhorz>7 & (A(1,1,:)~=3))+1e9;
G(ARvert>7 & (A(1,1,:)~=3))=G(ARvert>7 & (A(1,1,:)~=3))+1e9;

G(Deng>A(1,7,:).*(2))=G(Deng>A(1,7,:).*(2))+1e9;



%%Performance Restrictions
%Vmax Restrictions
%
 G((Vmax<Vstallalt))=1e9+G((Vmax<Vstallalt));%(-
Vmax((Vmax<Vstallalt))+Vstallalt((Vmax<Vstallalt))).^2;
% %Static Margin Constraints
 G((StaticMargin>SMpos))=100000+G((StaticMargin>SMpos));%+1000.*(-
StaticMargin((StaticMargin>SMpos))+SMpos).^2;
```

```
 G((StaticMargin<SMneg))=100000+G((StaticMargin<SMneg));%+1000.*(-
StaticMargin((StaticMargin<SMneg))+SMneg).^2;
% %Wing Loading Historical Guidelines
 G((W_S./9.81<minW_S))=1e9+G((W_S./9.81<minW_S));%+(-
W_S((W_S./9.81<minW_S))+minW_S).^2;
 G((W_S./9.81>maxW_S))=1e9+G((W_S./9.81>maxW_S));%+(-
W_S((W_S./9.81>maxW_S))+maxW_S).^2;
%
% %Thrust to Weight Historical Guidelines

G(((TorP_W.*9.81./1000)<0.07 &
A(1,18,:)~=3))=G((TorP_W.*9.81./1000<0.07 & A(1,18,:)~=3))+1000;%(-
TorP_W((TorP_W<0.07 & A(1,18,:)~=3))+0.07).^2;
G(((TorP_W.*9.81./1000)>1 & A(1,18,:)~=3))=G((TorP_W.*9.81./1000>1 &
A(1,18,:)~=3))+1e9;%(-TorP_W((TorP_W>0.50 & A(1,18,:)~=3))+0.50).^2;


G((TorP_W<0.25 & A(1,18,:)==3))=G((TorP_W<0.25 &
A(1,18,:)==3))+1000;%(-TorP_W((TorP_W<0.25 & A(1,18,:)==3))+0.25).^2;
G((TorP_W>1.0 & A(1,18,:)==3))=G((TorP_W>1.0 & A(1,18,:)==3))+1e9;%(-
TorP_W((TorP_W>1.0 & A(1,18,:)==3))+1.0).^2;

%Tail Coefficients Historical Guide
%Vertical cvt
G((cvt(1,1,:)<0.1 & A(1,1,:)~=3))=G((cvt(1,1,:)<0.1 &
A(1,1,:)~=3))+1000;%(-cvt((cvt(1,1,:)<0.02 & A(1,1,:)~=3))+0.02).^2;
G((cvt(1,1,:)>0.15 & A(1,1,:)~=3))=G((cvt(1,1,:)>0.15 &
A(1,1,:)~=3))+1000;%(-cvt((cvt(1,1,:)>0.15 & A(1,1,:)~=3))+0.15).^2;
%Horizontal cht
G((cht(1,1,:)<0.4 & A(1,1,:)~=3))=G((cht(1,1,:)<0.4 &
A(1,1,:)~=3))+1000;%(-cht((cht(1,1,:)<0.4 & A(1,1,:)~=3))+0.4).^2;
G((cht(1,1,:)>1 & A(1,1,:)~=3))=G((cht(1,1,:)>1 &
A(1,1,:)~=3))+1000;%(-cht((cht(1,1,:)>1 & A(1,1,:)~=3))+1).^2;
G((VmaxEndurance>VmaxRange))=G((VmaxEndurance>VmaxRange))+1e9;
G((VmaxRange>Vmax))=G((VmaxRange>Vmax))+1e9;

%Rate of Climb
G((RCmax<3))=G((RCmax<3))+1e9;
G((RCmaxalt<.508))=G((RCmaxalt<.508))+1e6;%(-
G((RCmaxalt<.508))+.508).^2;
%Takeoff and Landing Requirements
G((takeoff>runway))=G((takeoff>runway))+999;%(-
takeoff((takeoff>runway))+runway).^2;
G((landing>runway))=G((landing>runway))+999;%(-
landing((landing>runway))+runway).^2;



%Fuel Weight Ratio
G((Wfuel)>W.*2/3)=G((Wfuel)>W.*2/3)+1e9;
```

```
G((Wfuel)<W.*.15)=G((Wfuel)<W.*.15)+100000;
G(FuseL<=WingL)=G(FuseL<=WingL)+1e5;

% Mission Requirements
% G(Endurance<Endurancemin)=G(Endurance<Endurancemin)+1e9;
 G(Range>Rangemin+1000)=G(Range>Rangemin+1000)+1e9;

end
```

Function getCost

```
function [Cost,CPA]=getCost(A,Vmax,Wempty,a)

% Function uses the DAPCA IV Cost Model to estimate the cost of
producing Q
% aircraft in US Dollars (USD) Adjusted for 2016

Turbofantemp=1500;
Turboproptemp=1250;
Proptemp=273;
ind1=find(A(1,18,:)==1);
ind2=find(A(1,18,:)==2);
ind3=find(A(1,18,:)==3);
FTA=2;
Q=40;
REngineering=115.00;
RTooling=118.00;
RQuality=108.00;
RManufacturing=98.00;


We=Wempty.*9.81;
M=Vmax./a;
V=Vmax.*3.6;
He=5.18.*(We.^0.777).*(V.^0.894).*(Q.^0.163);       %Engineering hours
Ht=7.22.*(We.^0.777).*(V.^0.696).*(Q.^.263);        %Tooling Hours
Hm=10.5.*(We.^0.82).*(V.^0.484).*(Q.^.641);         %Manufacturing
Hours
Hq=0.133.*Hm;                                       %Quality Control
Hours
Cdev=67.4.*(We.^.630).*(V.^1.3);                    %Development
Support Cost
Cft=1947.*(We.^0.325).*(V.^0.822).*(FTA.^1.21);     %Flight Test Cost
Cm=31.2.*(We.^0.921).*(V.^0.621).*(Q.^0.799);       %Manufacturing
Materials Cost

Ceng(1,1,ind1(:))=1200.*A(1,19,ind1(:)).*(A(1,20,ind1(:)));
Ceng(1,1,ind2(:))=3112.*(9.66.*A(1,19,ind2(:)).*.8./Vmax(ind2(:))+M(ind
2(:)).*243.25+1.74.*Turboproptemp-2228);
```

```
Ceng(1,1,ind3(:))=3112.*(9.66.*A(1,19,ind3(:))+M(ind3(:)).*243.25+1.74.
*Turbofantemp-2228);

Cost=REngineering.*He+RTooling.*Ht+RManufacturing.*Hm+RQuality.*Hq+Cdev
(1,1,:)+Cft(1,1,:)+Cm(1,1,:)+Ceng(1,1,:).*A(1,19,:);
Cost=Cost.*1.05;                                         %Cost Avionics
Cost=Cost.*1.1;                                          %Cost Advanced
Materials
Cost=Cost.*1.048;                                        %Inflation Adjusted
CPA=Cost./Q;                                             %Cost per Aircraft
CPA(isinf(Cost))=1e12;
Cost(isinf(Cost))=1e12;
CPA(isnan(CPA))=1e12;
Cost(isnan(Cost))=1e12;
CPA(logical(imag(CPA)))=1e12;
Cost(logical(imag(Cost)))=1e12;
CPA(CPA<0)=1e12;
Cost(Cost<0)=1e12;
end
```

Function getK

```
function [K]=getK(AR,edih)
K(:,:,:)=1./(pi.*edih.*AR(:,:,:));
End
```

Function evaluate

```
function [O,P,G]=evaluate(AIRCRAFT)
rho=.75;
a=300;
[S,Sexp,Sfuse,Shorz,Svert,Sfusen,Sfusem,Sfuset]=getSs(AIRCRAFT);

[Swingwet,Shorzwet,Svertwet,Swet]=getSwet(Sexp,Shorz,Svert,Sfuse);%
[AR,AReff,ARvert,ARhorz]=getAspectRatio(AIRCRAFT,S,Svert,Shorz);%
[Weng,Leng,Deng,Heng,SFC,Dblade]=getEnginesize(AIRCRAFT);%
[VFuse,VFuel,TVolume,VWing,Vcut]=getVolumes(AIRCRAFT);%
clear Vfuse
clear Vwing
clear Vcut

[CDo,MACs,quarterwing,quarterhorz,quartervert]=getDragBuildup(AIRCRAFT,
S,Swingwet,Sfuse,Shorzwet,Svertwet,Dblade,Deng,rho);
[cgnosef,cgmainf,cgtailf,cgwing,qcwing,cgvert,cghorz,armhorz,armvert]=g
etCG(AIRCRAFT,MACs);
[cvt,cht]=getTailVolumeCoef(armhorz,armvert,AIRCRAFT,Svertwet,Shorzwet,
Sexp);
clear Swingwet
clear Shorzwet
clear Svertwet
clear Swet

[CLalpha]=getCLalpha(AIRCRAFT,AR,S,Sexp);
[edih]=getoswald(AIRCRAFT,AR,AReff,CLalpha);
[K]=getK(AR,edih);
[LDmax,CLmax]=getLDmax(K,CDo,CLalpha);
clear AReff

[W,Wempty,WFuel,Wwing,Whtail,Wvtail,Wengines,Wpayload,Wauxiliary]=getWe
ight(AIRCRAFT,Sexp,Sfuse,Shorz,Svert,VFuel,Weng);

clear Vfuel
[W_S,W_Sexp,TorP_W, TorP_Walt]=getW_SandTP_W(AIRCRAFT,W,S,Sexp,rho);



[Vmax,Vstall,Vtip,Vstallalt]=SteadyLevel(AIRCRAFT,S,W,CDo,K,CLmax,rho,T
orP_W,W_S,Dblade,a);
[VmaxRange,VmaxEndurance,Range,Endurance,RCmax,RCmaxalt]=getPerformance
(AIRCRAFT,WFuel,W,K,S,CDo,SFC,rho,Vmax,W_S,LDmax,TorP_W,TorP_Walt);
[MaxTurnV,MaxTurnRate,nMaxRate,MinTurnRadius,nmaxturn,nmaxneg]=
getTurn(rho,TorP_Walt,Vmax,VmaxRange,VmaxEndurance,CLmax,K,CDo,W_S,AIRC
RAFT);
[StaticMargin,CGx]=getStaticMargin(AIRCRAFT,cgnosef,cgmainf,cgtailf,cgw
ing,cgvert,cghorz,Sfusen,Sfusem,Sfuset,WFuel,Whtail,Wengines,Wpayload,W
auxiliary,Wwing,Wvtail,MACs,qcwing,W);

[takeoff,landing]=getRunway(AIRCRAFT,W_S,TorP_W,CLmax,Vstall,W,K,CDo,Se
xp);
[RCS,Abottom]=getRCS(AIRCRAFT,Sexp,Shorz,Svert,Dblade,Deng,Leng);
[Cost,CPA]=getCost(AIRCRAFT,Vmax,Wempty,a);
Range(isnan(Range))=0;
Endurance(isnan(Endurance))=0;
LDmax(logical(imag(LDmax)))=.1;
CPA(isnan(CPA))=1e9;
MinTurnRadius(isnan(MinTurnRadius))=5000000;
MinTurnRadius(logical(imag(MinTurnRadius)))=5000000;

[G]=getCPenalty(AIRCRAFT,AR,Vmax,Vstallalt,StaticMargin,W_S,TorP_W,cvt,
cht,RCmax,RCmaxalt,takeoff,landing,W,WFuel,Range,Endurance,Deng,a,VmaxE
ndurance,VmaxRange,ARvert,ARhorz);
%
[O,P]=getOBJValue(Vmax,Vstall,Vstallalt,StaticMargin,W_S,TorP_W,takeoff
,landing,W,WFuel,Range,Endurance,RCmax,nMaxRate,MaxTurnRate,MinTurnRadi
us,Cost,CPA,G,RCS,AR,LDmax);
end
```

```
function getlengthfuse


function [lfuse]= getLengthFuselage(A)
lfuse=A(1,8,:)+A(1,9,:)+A(1,10,:);
end



function getycgvortex


function [ycgv]=getycgvortex(A,Beta,AR)
taper=[1 .5 0.2 0.0];
a1m=[.5269 .4919 .5160 .5694];
a2m=[.123 .1413 .1176 .1202];
a3m=[.0441 .0157 0.0156 0.0083];
a4m=[-.0057 0.0054 -0.0023 -0.0028];
a5m=[0.0032 0.0061 0.0071 0.0081];
a1=interp1(taper,a1m,A(1,6,:),'pchip');
a2=interp1(taper,a2m,A(1,6,:),'pchip');
a3=interp1(taper,a3m,A(1,6,:),'pchip');
a4=interp1(taper,a4m,A(1,6,:),'pchip');
a5=interp1(taper,a5m,A(1,6,:),'pchip');
ycgv=a1+a2.*Beta.*AR+AR.*tan(A(1,5,:)).*(a3+a4.*Beta.*AR+a5.*AR.*tan(A(
1,5,:)));
end
```